FUJITSU

FUJITSU Software
Interstage AR Processing Server V1.1.1

Developer's Guide

B1WS-1107-02ENZ0(00)
December 2014

Preface

Purpose of this document

This document explains how to develop applications using Interstage AR Processing Server.

Intended readers

This document is intended for users who will design and develop applications using Interstage AR Processing Server. Readers of this document are also assumed to have knowledge of:

- Basic knowledge of Interstage AR Processing Server
- Web services
- Application server management
- Database management
- JavaScript
- JDK
- Application development using smart devices

Structure of this document

This document is structured as follows:

Chapter 1 Introduction

Provides an overview of the knowledge required to develop systems using Interstage AR Processing Server.

Chapter 2 Development flow

Explains how to develop using Interstage AR Processing Server.

Chapter 3 Creating 3D models

Explains the 3D models supported by Interstage AR Processing Server.

Chapter 4 Developing the overlay application (web application)

Explains how to develop the web application, which is a component of the overlay application.

Chapter 5 Developing the overlay application (native application)

Explains how to develop the native application, which is a component of the overlay application.

Chapter 6 Developing the system integration application

Explains how to develop the system integration application.

Chapter 7 Sample application (AR marker)

Explains how to run the sample applications that use AR marker recognition.

Chapter 8 Sample application (Location Data)

Explains how to run the sample applications specifically for use with Location Data and 3D models.

Chapter 9 Sample application (barcode)

Explains how to run the sample applications specifically for use with barcode.

Chapter 10 Sample application (beacon)

Explains how to run the sample applications specifically for use with beacon.

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
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 Introduction

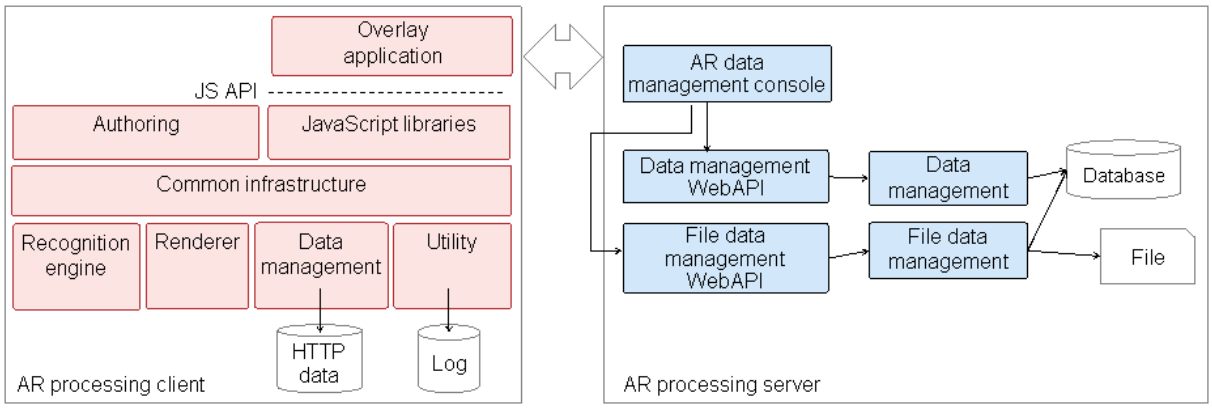
This chapter explains the architecture, data structure, and basic development flow that you should be familiar with before developing applications using Interstage AR Processing Server. This chapter is divided in the following sections:

- Architecture
  Explains the basic configuration of the features provided by AR processing clients and AR processing servers.
- AR overlay content structure
  Explains the information retrieved by an AR processing client from an AR processing server for displaying overlays, and the structure of AR overlay content.
- Data structure
  Data structure refers to the structure of the AR markers, scenarios, scenes, and AR overlay definitions stored in the database of Interstage AR Processing Server. This section provides the information required for using the overlay application and the system integration application to retrieve and update data stored in Interstage AR Processing Server.
- AR overlay content definition
  AR processing clients use AR overlay content definitions to render images and characters as AR overlay content. This section provides the information required for creating overlays using JavaScript, and for creating AR overlay content definitions without using the client authoring tool or the data management console.

1.1 Architecture

This section explains the architecture of Interstage AR Processing Server.

Figure 1.1 Architecture



AR processing client

This is the environment that runs on smart devices, for delivering image recognition AR. It overlays digital information such as images and characters retrieved from a server onto real-world camera images.

Recognition engine

Recognizes preview images taken by a camera. It then detects the ID number of the AR marker, and the distance and orientation of the AR marker in relation to the smart device.

Renderer

Analyzes the AR overlay definitions, and generates and renders AR overlay content.

Data management

Processes data and communications used by AR processing clients:

- Caches data used by the web layer, data used by the renderer, and files called by tap events to the file storage area in the smart device

- Provides a general-purpose viewing and searching interface for accessing the file storage area in the smart device (ContentProvider format)
- Communicates with the server (data upload and download)

Utility

Provides features common to AR processing clients, such as the ability to save and analyze external resource files and the ability to output logs.

Common infrastructure

Uses the components in an AR processing client to compose the overall structure and control the sequence, from camera recognition to rendering of the AR overlay content.

It has an internal web browser and connects to the JavaScript library to run processes on the browser.

Authoring

AR processing clients provide a feature that enables you to intuitively author (add and edit) AR overlay definitions on a smart device.

JavaScript library

Connects web applications and native applications. The AR processing client on Interstage AR Processing Server runs on Android, iOS, and Windows, so you can run the JavaScript library with the same interface on both types of terminals.

AR processing server

Centrally manages data in Interstage AR Processing Server, and manages AR overlay definitions so that they are overlaid appropriately for system users.

Data management and data management WebAPI

These components use an RDB to manage a range of information used by Interstage AR Processing Server, such as AR markers, scenarios, and AR overlay definitions.

File data management and file data management WebAPI

These components manage large-size data that cannot be managed by data management (file data such as images) and files that have no fixed format.

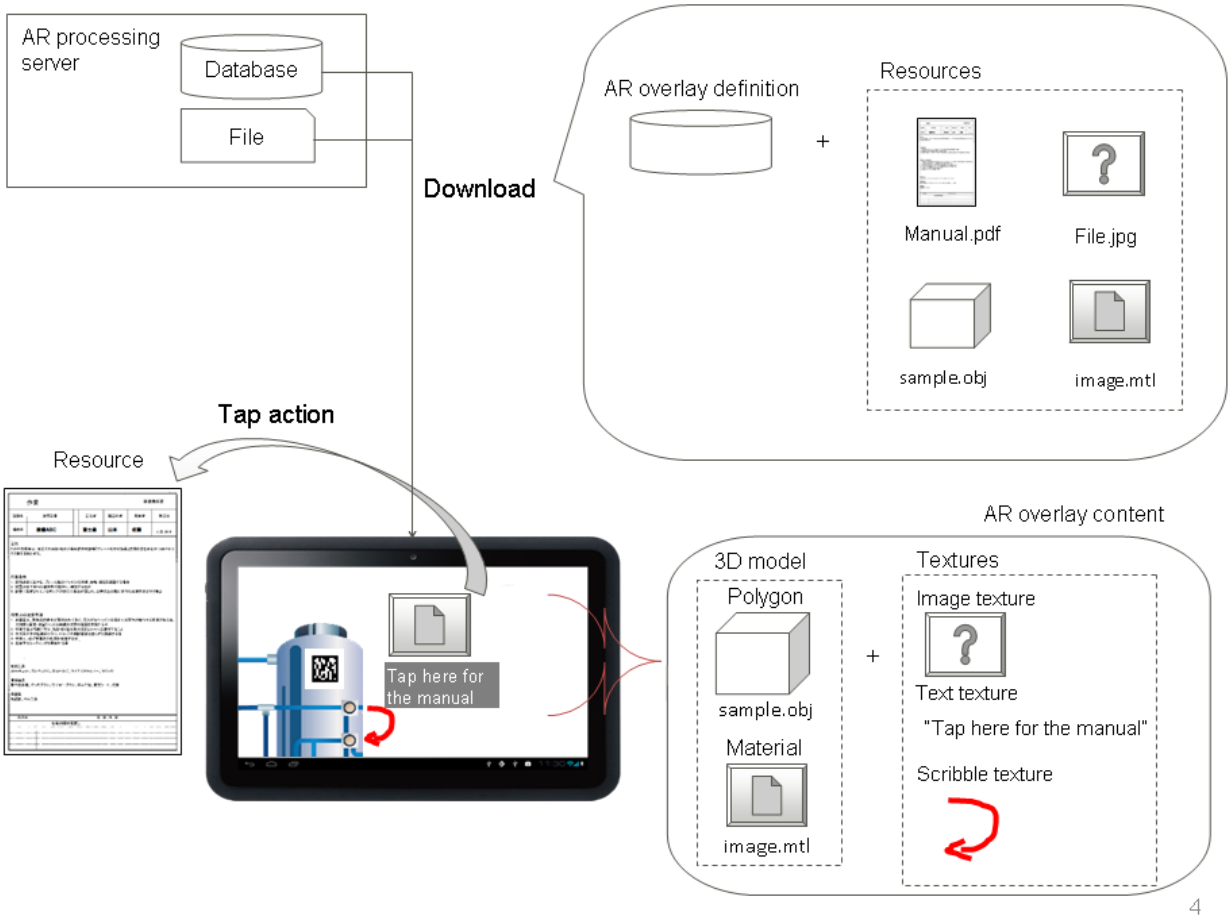
1.2 AR overlay content structure

The AR overlay content is made up of a 3D polygon (quadrilateral) and an image pasted onto a surface, known as a texture. There are three types of textures: Image textures, made up of an image file in a format such as JPG or PNG and retrieved as a resource Text textures, made up of character information Scribble textures, created using the scribble feature of the client authoring tool.

Interstage AR Processing Server retrieves the following information from the server, for rendering AR overlay content: The AR overlay content is rendered based on the retrieved information.

- AR overlay definition
  Information pre-associated with an AR marker to enable overlaying. The definition information indicates which AR overlay content is displayed for which AR marker, the size and rotation of display, and for which scenario and scene.
- Resource
  A file stored on an AR processing server or other media cloud, and used for tap action and AR overlay content.

Figure 1.2 Examples of AR overlay content



1.3 Data structure

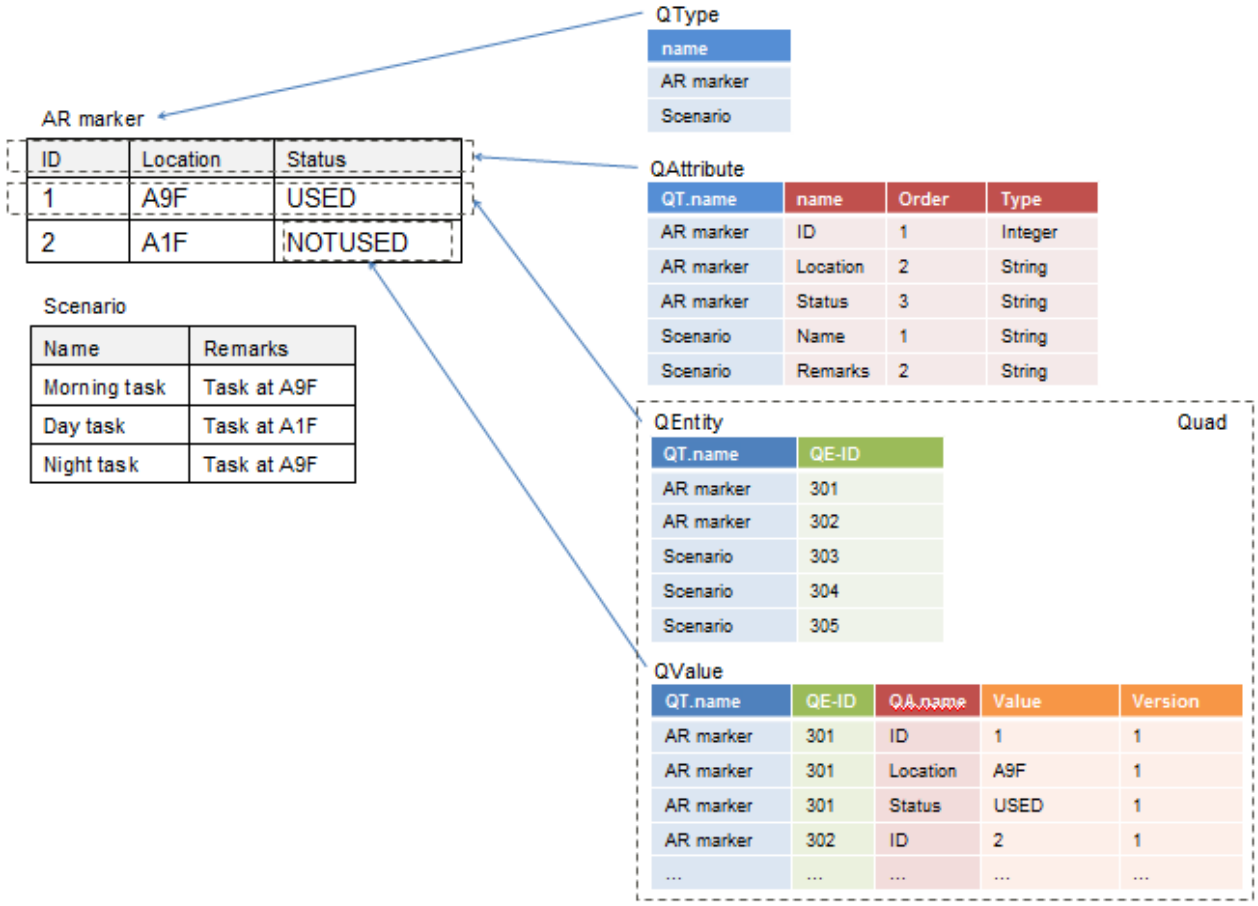
Interstage AR Processing Server associates multiple information items with a single AR marker. It uses AR markers and user permissions to search, sort, and connect only task-required information items for storing, retrieving, and editing them.

The EAV model used by Interstage AR Processing Server provides a framework that uses a relational database to manage data to perform these processes. The EAV model is a method of expressing data as a set with three elements: Entity (similar to row), Attribute (similar to column), and Value (similar to value). It can express various data structures.

1.3.1 Interstage AR Processing Server data structure

This section explains the data structure in Interstage AR Processing Server. All data stored by Interstage AR Processing Server is managed based on the EAV model - it is necessary to understand the EAV model below to use the REST API.

Figure 1.3 Data management diagram



- QType
  Stores 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 "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 ("value") using a conditional expression that is string-based rather than ID-based.

1.3.2 AR marker

The following table lists the members of the data type structure for managing the ID number, location, and state of an AR marker:

Table 1.1 AR marker: armk_fjarmarker

Attribute	Description	Data type
ar_id	AR marker ID	LONG
ar_idalias	Alias	STRING

Attribute	Description	Data type
ar_name	AR marker name	STRING
ar_description	Description	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG
armk_location	Location	STRING
armk_publishtime	Download time	LONG
armk_publishuser	Download user	STRING
armk_state	State - Unused - Used	STRING

1.3.3 Scenario

The following table lists the members of the scenario data type structure:

Table 1.2 Scenario: arscn_scenario

Attribute	Description	Data type
ar_id	Scenario ID	LONG
ar_name	Scenario name	STRING
ar_description	Description	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG

1.3.4 Scene

The following table lists the members of the scene data type structure:

Table 1.3 Scene: arsen_scene

Attribute	Description	Data type
ar_id	Scene ID	LONG
ar_name	Scene name	STRING
arscn_scenarioid	Scenario ID	LONG
ar_description	Description	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG

1.3.5 AR overlay definition

The following tables list the AR overlay definition data type structures provided by Interstage AR Processing Server and their attributes.

Table 1.4 AR overlay definition: arpoiarmk_default

Attribute	Description	Data type
ar_id	AR overlay definition ID	LONG
ar_name	AR overlay definition name	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG

Attribute	Description	Data type
arscn_scenarioid	Scenario ID	LONG
arsen_sceneid	Scene ID	LONG
armk_markerid	AR marker ID	LONG
arpoi_superimposedgraphic	AR overlay content definition	STRING

Table 1.5 AR overlay definition: arpoigps_default

Attribute	Description	Data type
ar_id	AR overlay definition ID	LONG
ar_name	AR overlay definition name	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG
arscn_scenarioid	Scenario ID	LONG
arsen_sceneid	Scene ID	LONG
ar_base	AR base This is hard-coded to "GPS".	STRING
arpoi_superimposedgraphic	AR overlay content definition	STRING

Table 1.6 AR overlay definition: arpoibarcode_default

Attribute	Description	Data type
ar_id	AR overlay definition ID	LONG
ar_name	AR overlay definition name	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG
arscn_scenarioid	Scenario ID	LONG
arsen_sceneid	Scene ID	LONG
ar_base	AR base This is hard-coded to "BARCODE".	STRING
arbarcode_content	Barcode ID	STRING
arpoi_superimposedgraphic	AR overlay content definition	STRING

Table 1.7 AR overlay definition: arpoibeacon_default

Attribute	Description	Data type
ar_id	AR overlay definition ID	LONG
ar_name	AR overlay definition name	STRING
ar_registrationtime	Registration datetime	LONG
ar_modificationtime	Update datetime	LONG
arscn_scenarioid	Scenario ID	LONG
arsen_sceneid	Scene ID	LONG
ar_base	AR base This is hard-coded to "BEACON".	STRING
arbeacon_uuid	Beacon UUID	STRING
arpoi_superimposedgraphic	AR overlay content definition	STRING

1.4 AR overlay content definition

This section explains the structure of the AR overlay content definition contained in an AR overlay definition. The AR overlay content definition (arpoi_superimposedgraphic) is specified in JSON format.

Using server authoring or client authoring automatically generates an AR overlay content definition, but the system developer can also create one using JavaScript.

Example

The following examples show basic descriptions.

- To overlay the image http://*server*:*portNumber*/*filePath*/sample.png

```
{"typeName":"SuperimposedGraphic",
 "projectionType":"PERSPECTIVE",
 "graphic":{"typeName":"SquareModelGraphic",
   "texture":{"typeName":"ImageTexture",
   "src":"http://server:portNumber/filePath/sample.png"},
   "scale":{"typeName":"Point","x":1,"y":1,"z":1}},
   "translation":{"typeName":"Point","x":23,"y":20,"z":1}
}
```

- To overlay the string "sample"

```
{"typeName":"SuperimposedGraphic",
 "projectionType":"ORTHO2D",
 "translation":{
   "typeName":"Point","z":0,"y":0,"x":0},
 "rotation":{
   "typeName":"Point","z":0,"y":0,"x":0},
 "graphic":{
     "typeName":"SquareModelGraphic",
     "scale":{"typeName":"Point","z":1,"y":1,"x":1},
     "texture":{"typeName":"TextTexture",
       "text":"sample",
       "fontSize":50,
       "color":"4278190335",
       "wordWrap":false}}
}
```

- To overlay the image http://*server*:*portNumber*/*filePath*/sample.png that calls a separate file http://*server*:*portNumber*/*filePath*/sample.pdf when tapped

```
{"typeName":"SuperimposedGraphic",
 "projectionType":"PERSPECTIVE",
 "graphic":{"typeName":"SquareModelGraphic",
   "texture":{"typeName":"ImageTexture",
    "src":" http:// server:portNumber/filePath/sample.png"},
    "scale":{"typeName":"Point","x":1,"y":1,"z":1}},
    "translation":{"typeName":"Point", "x":-1,"y":6,"z":1},
    "tapAction":{"typeName":"URLAction",
    "src":" http:// http://server:portNumber/filePath/sample.pdf"}
}
```

- To overlay the string "sample" and have it call the method Apl.sample() when tapped

```
{"typeName":"SuperimposedGraphic",
 "projectionType":"ORTHO2D",
 "translation":{
   "typeName":"Point","z":0,"y":0,"x":0},
 "rotation":{
   "typeName":"Point","z":0,"y":0,"x":0},
 "graphic":{
```

```
   "typeName":"SquareModelGraphic",
   "scale":{"typeName":"Point","z":1,"y":1,"x":1},
   "texture":{"typeName":"TextTexture",
     "text":"sample",
     "fontSize":50,
     "color":"4278190335",
     "wordWrap":false}}
   "tapAction":{"typeName":"ScriptAction",
   "expression":"Apl.sample()"}
}
```

1.4.1 Overview of arpoi_superimposedgraphic

The following tables provide an overview of all elements referred in the samples above.

Table 1.8 typeName

typeName	Description	Type
SuperimposedGraphic	AR overlay content type	SuperimposedGraphic

Table 1.9 SuperimposedGraphic member list

Member	Description	Type
translation	- AR marker/Barcode Relative distance from the AR marker or Barcode - Location Data Latitude, longitude, and altitude of AR overlay content - Beacon Relative distance from the center of the smart device screen	Point
rotation	- AR marker Rotational angle relative to the AR marker - Location Data Rotational angle based on the orientation and horizontal direction - Beacon/Barcode Absolute rotational angle	Point
projectionType	2D/3D information about the AR overlay content (required key)	ProjectionType
tapAction	Information about action to perform when AR overlay content is tapped	ActionHolder
graphic	AR overlay content information (required key)	SquareModelGraphic

1.4.2 translation

- If using AR markers, barcodes or beacons

  Contains the relative distance between the AR overlay content and the AR marker or barcode.
  If using beacons, contains the ratio from the center of the smart device screen.

  Details of this member are as follows:

  - Point

    - x: Indicates the value in the x-axis direction. 1=length of the AR marker
      Type: float,
      Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).

Range: -32 to 32

- y: Indicates the value in the y-axis direction. 1=length of the AR marker
  Type: float,
  Precision: 3 decimal places (values with more decimal places, will be rounded to 3 decimal places).
  Range: -32 to 32

- z: Indicates the value in the z-axis direction. 1=length of the AR marker
  Type: float,
  Precision: 3 decimal places (values with more decimal places, will be rounded to 3 decimal places).
  Range: -32 to 32

Table 1.10 Translation point units

Recognition mode	Standard value of 1 point
AR marker	Length of the AR marker
Barcode	Long side of the barcode
Beacon	50 [dip]

- If using Location Data

Contains position information for the AR overlay content.

Details of this member are as follows:

- Point

  - Latitude: Indicates the latitude. Use positive values for north latitudes and negative values for south latitudes. The unit is degrees.
    Type: float,
    Precision: 5 decimal places (set in units of meters).
    Range: -90 to 90

  - Longitude: Indicates the longitude. Use positive values for east longitudes and negative values for west longitudes. The unit is degrees.
    Type: float,
    Precision: 5 decimal places (set in units of meters).
    Range: -180 to 180

  - Altitude: Indicates the value in the z-axis direction. The unit is meters.
    Type: float,
    Stride: Values set in unit of meters are valid.
    Range: -100,000 to 100,000

Example

- To display AR overlay content on the AR marker

```
"translation":{"typeName":"Point","x":0,"y":0,"z":0}
```

1.4.3 rotation

- If using AR markers, barcodes or beacons

Contains the rotational angle relative to the AR marker. It is effective only for the z-axis for AR overlay content displayed in 2D. If using barcodes or beacons, contains the absolute rotational angle. Details of this member are as follows:

- Point

  - x: Indicates the value in the x-axis direction. The unit is degrees.
    Type: float,
    Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
    Range: -360 to 360

- y: Indicates the value in the y-axis direction. The unit is degrees.
  Type: float,
  Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
  Range: -360 to 360
- z: Indicates the value in the z-axis direction. The unit is degrees.
  Type: float,
  Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
  Range: -360 to 360

- If using Location Data

  Contains the rotational angle based on the orientation and horizontal direction. It is effective only for the z-axis for AR overlay content displayed in 2D. Details of this member are as follows:

  - Point
    - x: Indicates the value in the horizontal direction. The unit is degrees.
      Type: float,
      Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
      Range: -360 to 360
    - y: Indicates the value in the orientation direction. North is set as 0. The unit is degrees.
      Type: float,
      Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
      Range: -360 to 360
    - z: Indicates the value in the z-axis direction. The unit is degrees.
      Type: float,
      Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
      Range: -360 to 360

Example

- To display AR overlay content without rotating it

```
" rotation":{"typeName":"Point","x":0,"y":0,"z":0}
```

1.4.4 projectionType

Contains the AR overlay content projection method.

Details of this member are as follows:

- projectionType
  - PERSPECTIVE: Displays AbstractBuiltinModelGraphic in 3D
  - ORTHO2D: Displays AbstractBuiltinModelGraphic in 2D

Display size of AR overlay content by ProjectionType (if using AR marker or Barcode)

projectionType	Size of AR overlay content [pixel]	Marker size			
		50 [mm]	100 [mm]	200 [mm]	400 [mm]
ORTHO2D	32	50 [pixels]	50 [pixels]	50 [pixels]	50 [pixels]
	1024	1600 [pixels]	1600 [pixels]	1600 [pixels]	1600 [pixels]
PERSPECTIVE	32	50 [pixels]	100 [pixels]	200 [pixels]	400 [pixels]
	1024	1600 [pixels]	3200 [pixels]	6400 [pixels]	12800 [pixels]

Display size of AR overlay content by ProjectionType (if using Location Data)

projectionType	Size of AR overlay content [pixel]	Distance from content			
		20 [meters]	100 [meters]	500 [meters]	1000 [meters]
ORTHO2D	32	50 [pixels]	50 [pixels]	50 [pixels]	50 [pixels]
	1024	1600 [pixels]	1600 [pixels]	1600 [pixels]	1600 [pixels]
PERSPECTIVE	32	50 [pixels]	10 [pixels]	2 [pixels]	1 [pixel]
	1024	1600 [pixels]	320 [pixels]	64 [pixels]	32 [pixels]

Display size of AR overlay content by ProjectionType (if using Beacon)

- If using beacons, the size if fixed, because the distance from the beacon is not detected.

projectionType	Size of the overlay content [pixel]	Beacon recognition
ORTHO2D	32	50 [pixels]
PERSPECTIVE	32	- (not supported)

Example

- To display AR overlay content in 3D

```
"projectionType":"PERSPECTIVE"
```

1.4.5 tapAction

Contains tap action information for AR overlay content.

Details of this member are as follows:

- ActionHolder
  - URLAction: Indicates the URL specified when the screen is tapped
    - src: URL of information displayed when the screen is tapped Enter this value in 256 characters or less.
      Type: URL
  - ScriptAction: Calls a JavaScript method when the screen is tapped
    - expression: Name of the JavaScript method called when the screen is tapped Enter this value in 256 characters or less.
      Type: String

Example

- To call another file when the screen is tapped

```
"tapAction":{"typeName":"URLAction","src":"fileUrl"}
```

- To call a JavaScript method when the screen is tapped

```
"tapAction":{"typeName":"ScriptAction","expression":"methodName"}
```

1.4.6 graphic

Contains AR overlay content information.

Table 1.11 typeName

typeName	Description	Type
SquareModelGraphic	Specifies SquareModel	SquareModelGraphic

Table 1.12 Member list

Member	Description	Type
scale	AR overlay content magnification/reduction rate	Point
texture	Texture of AR overlay content	AbstractTexture

Details of this member are as follows:

- If using AR markers, barcodes or beacons
  - scale: Defines the magnification/reduction rate of AR overlay content
    - Point
      - x: Indicates the relative size in the x-axis direction. 1=Length of the frame of the AR marker
        Type: float,
        Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
        Range: 0.1 to 32 times
      - y: Indicates the relative size in the y-axis direction. 1=Length of the frame of the AR marker
        Type: float,
        Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
        Range: 0.1 to 32 times
      - z: Indicates the relative size in the z-axis direction. 1=Length of the frame of the AR marker
        Type: float,
        Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
        Range: 0.1 to 32 times
  - texture: Overlays characters, an image, or scribble
    - ImageTexture: Image
    - TextTexture: Characters
    - HandwritingTexture: Scribble
- If using Location Data
  - scale: Defines the magnification/reduction rate of AR overlay content
    - Point
      - x: Indicates the relative size in the x-axis direction. 1=Length of the AR content
        Type: float,
        Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
        Range: 0.1 to 150,000 times
      - y: Indicates the relative size in the y-axis direction. 1=Length of the AR content
        Type: float,
        Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
        Range: 0.1 to 150,000 times
      - z: Indicates the relative size in the z-axis direction. 1=Length of the AR content
        Type: float,
        Precision: 3 decimal places (values with more decimal places will be rounded to 3 decimal places).
        Range: 0.1 to 150,000 times
  - texture: Overlays characters, an image, or scribble
    - ImageTexture: Image
    - TextTexture: Characters
    - HandwritingTexture: Scribble

1.4.6.1 ImageTexture

Sets texture information created for an image.

Table 1.13 typeName

typeName	Description	Type
ImageTexture	Specifies ImageTexture	ImageTexture

Table 1.14 Member list

Member	Description	Type
src	Path of resource used for texture (required key). The maximum resource size that can be used for the image texture is 1024 [pixels] x 1024 [pixels]. The recommended size is 5 MB.	URL (supported URLs: http(s))

Details of this member are as follows:

- src: Path of the resource to be used for the image texture
  - Type: URL
  - Description: Enter the URL of the resource to be used for the image texture in 256 characters or less.
  - Specifiable file formats:
    - png
    - jpg, jpeg

1.4.6.2 TextTexture

Sets texture information created for a string.

Table 1.15 typeName

typeName	Description	Type
TextTexture	Specifies TextTexture	TextTexture

Table 1.16 Member list

Member	Description	Type
backgroundColor	Background color. The value of ARGB is expressed in decimal.	int
size	Texture rendering size (in [pixels])	Size
text	Characters to be displayed. This is a required key. To enter the symbols ", /, and \, type \", \/, and \\ respectively in accordance with the JSON format escape definition.	String
fontSize	Font size of the string (in [pixels])	int
color	Font color. The value of ARGB is expressed in decimal.	int
wordWrap	Text wrapping enabled/disabled (not reflected if no size is specified)	boolean

Details of this member are as follows:

- backgroundColor: Background color. The value of ARGB is expressed in decimal.
  - Type: int
  - Range: 0 to 4294967295

  Example

  Non-transparent blue:

    - ARGB: FF00FF00
    - int type: 4278255360

- size: Texture rendering size
    - typeName: Size
        - width: Texture width. Enter a value in the range 10 to 1024 [pixels].
        - height: Texture height. Enter a value in the range 10 to 1024 [pixels].
- text: Characters to be displayed
    - Type: String
    - Range: Enter 1 to 256 characters.
- fontSize: Size of the characters
    - Type: int
    - Range: 10 to 750
      By default, strings are displayed at 40 [pixels].
- color: Font color. The value of ARGB is expressed in decimal.
    - Type: int
    - Range: 0 to 4294967295

Note

- You must specify the size if using wordWrap.
- Do not specify the size if not using wordWrap. If the display area for entered characters is larger than the area specified in size, not all characters may be displayed.
- When a string texture is rendered as AR overlay content with wordWrap disabled, a margin is displayed above and below the characters. The height of the texture is therefore the texture size which is greater than or equal to the value set for fontSize (fontSize: height approximately equal to 3:4).

Example

- To display "Sample" with wordWrap enabled

```
"texture" : {
"typeName": "TextTexture",
"backgroundColor": 4294967295,
"size": {
"typeName" : "size",
"width" : 64,
"height" : 64
},
"text" : "Sample",
"fontSize" : 20,
"wordWrap" : true
}
```

1.4.6.3 HandwritingTexture

Sets image texture information created by the scribble feature provided by the client authoring tool.

Table 1.17 typeName

typeName	Description	Type
HandwritingTexture	Specifies HandwritingTexture	HandwritingTexture

Table 1.18 Member list

Member	Description	Type
src	Path of resource used for texture (required key). The maximum resource size that can be used for image texture is 1024 [pixels] x 1024 [pixels]. The recommended size is 5 MB.	URL (supported URLs: http(s))
handwritingColor	Scribble color	

Details of this member are as follows:

- src: URL of the image texture
  - Type: URL
  - Range: Enter the URL of the resource to be used for the image texture in 256 characters or less.
  - Specifiable file formats:
    - png
    - jpg, jpeg
- handwritingColor: Scribble color. The value of ARGB is expressed in decimal.
  - Type: int
  - Range: 0 to 4294967295

Chapter 2 Development flow

This chapter explains how to develop applications using Interstage AR Processing Server, using the provided sample application (AR marker) as an example.

The basic flow for developing applications using Interstage AR Processing Server is as follows:

1. Planning
   Examine the business and jobs, identify jobs for applying AR, and decide how to use it.
2. Defining requirements
   Verify requirements for systemization of the jobs identified for applying AR in the planning stage, define the functional requirements, and design an overview of the system.
3. Designing the system
   Design the external specifications (such as system features and interfaces) of the system from the viewpoint of the users. Also design the internal structure of the system, based on the external specifications.
4. Creating the system
   Program and compile the overlay application and the system integration application according to the design, and create a deployment file. Also create resources to be used for displaying overlays.

See

Refer to the *Operator's Guide* for details on authoring and how to deploy the applications you create.

2.1 Planning

Interstage AR Processing Server overlays various types of digital information such as images and text onto real-world objects. It is therefore very versatile and can be used in all business areas and for all jobs.

The planning stage prior to using Interstage AR Processing Server involves examining the business and jobs, identifying those suitable for applying AR, and deciding how to use it. If you are planning a system that uses AR, consider the following in addition to the usual planning:

- Is there anyone to use the displayed content?
  AR technology is intended to assist humans, so it is not suited to fully automated tasks.
- Can smart devices be used?
  Smart devices are used as the AR processing clients, so they must be brought into the workplace.

Example

Table 2.1 Examples of business considerations

No	Business operation	Business challenge	AR-based measure	Applicability
1	Manufacturing	Save time	Lighten the burden on workers by providing appropriate work instructions	Lower the priority and exclude at this time, because automation is underway
		Improve product value	Use AR for manuals provided to customers	
		Prevent accidents during manufacturing	Prevent accidents by providing precautions before work starts and by presenting past examples	
2	Inspection	Improve inspection accuracy	Prevent inspection omissions by overlaying inspection items onto a product	Excluded from the current systemization due to the low cost-effectiveness
		Reduce inspection time	Lighten the burden on workers by providing appropriate work instructions	

No	Business operation	Business challenge	AR-based measure	Applicability
3	Maintenance	Prevent accidents during inspection	Prevent accidents by providing precautions before work starts and by presenting past examples	Include in the current systemization
		Reduce man-hours	Lighten the burden on workers by providing appropriate work instructions	
4	Plant inspection	Improve customer value	Aim to increase visitors by providing AR overlay content to visitors to the plant	Include in the current systemization

2.2 Defining requirements

Before using Interstage AR Processing Server, consider the following in addition to the usual requirements.

2.2.1 Considering recognition modes and content

Interstage AR Processing Server can display various overlay contents by using recognition information such as AR markers and Location Data. Therefore, consider what content you want to display for which recognition information according to recognized targets.

- Which recognition mode do you want to use for a job?
  Consider which recognition mode (or modes) provided by Interstage AR Processing Server that you want to use for your job.
- Can AR markers be affixed at the site, and if so, where will they be affixed?
  If using AR marker recognition mode, consider if there is anywhere to affix AR markers to objects onto which overlays are to be displayed, and whether users can hold a smart device up to those objects.
- What content do you want to overlay?
  Consider what content you want to display immediately for the user or to superimpose on an object, and whether data exists.

Example

Table 2.2 Examples of determining target tasks

No	Target task	Task description	AR target	Content

1	Temperature inspection	Inspect, twice daily (morning and night), the hydraulic presses installed at a total of 8 locations within each building to ensure there is no problem with pressure, flow rate, or temperature.	- Water heater (AR marker)
			  - Valves
			  - Interior of the equipment
			  - Thermometer
				- Work sequence
				- Input field
				- Contact information
				- Previous inspection results

2	Facility guidance	Guide visitors who have come to tour the factory.	- Building 1 (location data) - Building 2 (location data)	Facility guidance

2.2.2 Size and material of AR markers

You must adjust the AR markers for Interstage AR Processing Server to suit the environment where they are used.

When considering the size of an AR marker, take the following into account (refer to the *Operator's Guide* for details on the relationship between the AR marker and the recognition distance).

- The distance between the position from which the user will hold up the smart device and the AR marker affixed to the target object
- How to print the AR markers

- The performance of the smart device to be used

Example

Examples of locations and AR marker sizes

No	Installed target	Location	Distance from the user	Content	AR marker size	Print medium
1	Water heater A	Control room in the basement of East wing A	Approx. 1 to 1.5 m	- Work sequence - Input field - Contact information - Previous inspection results	Default	Print using a label printer
2	Water heater B	Control room in the basement of East wing A	Approx. 1 to 1.5 m	- Manual - Work sequence	Default	Print using a label printer

2.2.3 Considering Location Data

If using Location Data on Interstage AR Processing Server, you must adjust the information to suit the environment where it is used.

When considering Location Data, take the points below into account. Refer to the *Operator's Guide* for details.

- How will the Location Data be retrieved?
  Consider how the Location Data will be retrieved in light of, for example, the accuracy required for the job or the preparation of facilities needed to use it.
- What will be displayed from the location to be used?
  Consider the object display range and how much content will be displayed from the location that the smart device is held up to.
- What content do you want to overlay?
  Consider what content you want to display immediately for the user or to superimpose on an object, and whether data exists.

2.2.4 Format, size and material of barcodes

If using an existing barcode, you must ensure that the format is supported. Refer to the *Operator's Guide* for details on formats. Refer to "Size and material of AR markers" for details on considerations regarding size and material.

2.2.5 Considering beacons

If using beacons on Interstage AR Processing Server, you must adjust the information to suit the environment where they are used. When considering beacons, take the points below into account. Refer to the *Operator's Guide* for details.

- Data format
  Consider if any specific information should be appended to the data sent by the beacons. You can use the application to analyze the information, so that it operates in accordance with the format.
- UUID
  Specify the UUID to be set for the beacon. The UUID is required to recognize the beacon.
- What are you performing beacon recognition for?
  Are you planning to perform content overlay, or will you use it in combination with another recognized target?

Note

The beacon recognition mode is supported only in Android.

2.2.6 Considering the system configuration

When contemplating the system configuration of Interstage AR Processing Server, consider the following in addition to the usual system configuration (refer to the *Overview* for details on the environment and how to estimate the disk capacity).

- Configuration pattern
  Interstage AR Processing Server enables you to select the configuration pattern to suit the environment, the number of concurrent user accesses, the volume of contexts to be handled, and security.
- Communication with the server
  Interstage AR Processing Server uses a server to centrally manage information, but once data is downloaded, a site can operate offline. Therefore, consider whether the site has a communication environment. Even if it does not, consider before visiting the site whether it has an environment that would enable data to be retrieved from a server.
- Retrieval of Location Data

  If using Location Data recognition mode, consider a communications agreement, such as with a mobile phone provider.
- Smart device selection
  Interstage AR Processing Server uses a smart device to display overlays. Consider the type and quantity of smart devices to be used, taking the following into account:
  - Required recognition distance
  - Usage and recharge times per task
  - Required memory capacity
- Database capacity estimation
  The database capacity required for Interstage AR Processing Server is calculated based on the number of scenarios, the average number of scenes per scenario, the number of AR overlay definitions used per scene, and resource files. Refer to the *Overview* for details on the formula.

2.3 Designing the system

Before using Interstage AR Processing Server, consider the following in addition to the usual design elements:

- Security risk
  Consider the security risk inherent in using the system.
- Design of data items
  If integrating with an existing system, consider what data is to be viewed, registered, and stored, and at what time.
- Responsive design
  If you want to use different types of devices such as smart phones and tablets for viewing the web applications you develop, consider responsive design.
- Scenario and scene design
  Consider the scenarios and scenes to be used in the system.
- AR overlay definition design
  Design the information required for authoring.
- User-defined table design
  Design the tables for storing external data to be used by the overlay application.
- External design
  Design the external specifications of the system from the viewpoint of the system users.
- Internal design
  Design the internal structure of the system and the program interfaces based on the external specifications.

2.3.1 Considerations for reducing security risk

Consider the following points for reducing security risk while using Interstage AR Processing Server.

2.3.1.1 Overlay application (web application)

This product uses WebView to provide HTML rendering and a JavaScript execution feature. WebView entails a security risk due to the nature of the content being accessed. In particular, executing an invalid JavaScript code may result in a security breach, such as a third party operating the device or acquiring information from it.

Ensure that web applications display content from your own administration server on the Internet.

The following lists possible measures:

- When setting the URL for starting the overlay application, set a fixed URL that links the application to the content in your administration server (do not allow system users to specify the URL).
- Use HTTPS as the communication protocol for downloading the overlay application.

 Example

Specifying the URL for calling the overlay application

```
//Set the overlay application Activity to intent.
String _package = "com.fujitsu.interstage.ar.mobileclient.android";
String _activity = "com.fujitsu.interstage.ar.mobileclient.android.base.web.ArWebMainActivity";
Intent intent = new Intent();
intent.setClassName(_package, _activity);
//1) Set a fixed URL from the application as the activation URL
//2) Use HTTPS as the communication protocol
intent.putExtra("extraskey_widget_url", "https://www.fujitsu.com/index.html");
//Start Activity.
startActivity(intent);
```

2.3.1.2 Built-in storage

The files (offline resources, photos taken using authoring, log files) to be used by the applications are stored in built-in storage. You can view their content from a PC via a USB connection. The files are deleted when, for example, the application is uninstalled. If you do not want data to remain in the device, the system users should delete it using File Explorer.

2.3.2 Designing data items

You must consider the following if you want to integrate with a core business system:

- Data items to be reflected from the AR processing server to the core business system
  If, for example, you want to view previously stored data such as a history on site, adjust the data to be used with the overlay application.
- Data items to be reflected from the AR processing server to the core business system
  Consider what data is to be transferred when, for example, values entered by a user in the overlay application are to be reflected in the core business system.
- Managing data on AR Processing Server
  Consider how to manage data to suit your business style, such as whether to manage it using a user-defined table or to use a media cloud.
- Timing for reflecting data
  Consider the timing for reflecting data from the core business system and Interstage AR Processing Server.
- Appending position information and a recognition mode for displaying content (for displaying data from a core business system as AR overlay content)
  There are two methods of creating an AR overlay definition: creating one from scratch by using a JavaScript library, and converting data already registered in the AR processing server. Consider which method to use for appending a recognition mode and position information.

Point

If you integrate with multiple core business systems, there may be duplicate parameter names and values. If this happens, problems may arise if a core business system application or AR business application retrieves or updates the wrong data. To avoid such problems, you must take measures to prevent duplication by, for example, attaching a prefix to the data managed by the AR processing server to identify each core business system.

2.3.3 Considering responsive design

Depending on your business, you might use several different types of smart devices, such as smartphones and tablets, rather than a single type as your AR processing client. In this case, consider responsive design.

2.3.3.1 Designing web applications

You must ensure that the screen layout of web applications is displayed properly even if smart devices with different screen sizes are used. In this case, consider responsive design for leveraging the single source multi-platform approach, instead of creating HTML/CSS for each screen size.

However, in some cases different web applications may be provided depending on the job purpose. For example, tablets may be used for jobs where a large screen area is needed for entering characters and reading manuals, and smartphones may be used for jobs that do not require a large screen. In this situation, consider whether responsive design would be appropriate.

2.3.3.2 Overlaying

If using smart devices with different screen sizes and resolutions to display overlays based on a single AR overlay definition, consider the following:

- Distance between the AR marker and the smart device
  To display overlays from Interstage AR Processing Server, you first calculate the position for placing AR overlay content based on the size of the AR marker captured by the camera. Therefore, differences in resolution or screen size never cause the overlays to overlap or be displayed in the wrong place. Ensure that the position where you want to display AR overlay content will be within the frame of the camera when the smart device is held up to the AR marker from the expected position.
- Displaying AR overlay definition content with 2D projection
  When the projection is 2D, AR overlay content is rendered according to the specified size in pixels. Therefore, when the same content is displayed using smart devices with different resolutions, it may appear differently on the different smart devices. For example, it may be displayed in full on smart devices with a high resolution but be cut off on smart devices with a low resolution. Consider the size of AR overlay content when displaying it in 2D.

2.3.4 Designing scenarios and scenes

Interstage AR Processing Server uses scenarios and scenes for switching AR overlay content. Therefore, for each scenario and scene, design what you want to overlay on which AR marker and what you want to instruct the user to do, using your requirement definition as a basis.

Example

Example scenario and scene design

Scenario		Scene		Operation content
Scenario name	Scenario ID	Scene name	Scene ID	
Inspection work	1	Operate valves	1	Display manual Provide work instructions
		Check interior	2	Play video
		Check value	3	Provide work instructions

Scenario		Scene		Operation content
Scenario name	Scenario ID	Scene name	Scene ID	
		Enter temperature	4	Input temperature

2.3.5 Designing AR overlay definitions

Design the overlays to be used with the overlay application. Consider the following:

- Basic information
  - Overlay name
    Specify the name of the defined overlay. Specifying a unique name will make it easier to search for the overlay from an application and to specify the overlay.
  - Scenario ID
    ID of the scenario to be overlaid.
  - Scene ID
    ID of the scene to be overlaid.
  - Recognition mode
    Select **AR marker**, **Barcode**, **Location Data**, or **Beacon**.
    If **AR marker** is selected, then the ID of the AR marker over which the overlay content is to be superimposed must also be specified.
    If **Barcode** is selected, then the ID of the barcode over which the overlay content is to be superimposed must also be specified.
    If **Location Data** is selected, then the latitude, longitude, and altitude of the location over with the overlay content is to be superimposed must also be specified.
    If **Beacon** is specified, then the UUID of the beacon that will cause the overlay content to be superimposed must also be specified.
- Overlay definition content
  Select from the following types:
  - Text
    Use this type to overlay text information.
  - Shapes
    Select this type to use the product-standard shapes.
  - File
    You can use any image file as an overlay. Specify a file saved using file management on the AR processing server or the URL of a file located externally. The overlay is displayed when the network is accessed for downloading the overlay image.
  - 3D model
    Use this type to overlay a 3D model. A 3D model is a data structure for representing three-dimensional objects on a display. It is created by analyzing the obj and mtl files explained in Chapter 3 Creating 3D models.
- Overlay method
  - Projection
    Select 2D display or 3D display. 2D display is a parallel projection of AR content, and 3D display is a perspective projection of AR content. Note that 3D models are supported only in 3D display.

Note

If the recognition mode is "Beacon", overlay content can be displayed only in 2D mode.

  - If using AR markers
    - Relative position from the AR marker
      Specify the x-, y-, and z-coordinates from the center of the AR marker, using one side of the AR marker as the measurement unit.
    - Rotational angle relative to the AR marker

Specify the rotational angle of the x-, y-, and z-coordinates relative to the AR marker coordinate system, using the center of the AR marker as the reference point.

- If using barcodes
  - Relative position from the barcode
    Specify the x-, y-, and z-coordinates from the center of the barcode, using the long side of the barcode as the measurement unit.
  - Rotational angle relative to the barcode
    Specify the rotational angle of the x-, y-, and z-coordinates relative to the barcode coordinate system, using the center of the barcode as the reference point.
- If using Location Data
  - Location Data
    Specify Location Data using the latitude, longitude, and altitude parameters.
  - Rotational angle
    Specify the rotational angle of rotation based on the orientation and horizontal direction. It is effective only for the z-axis for AR overlay content displayed in 2D.
- If using beacons
  - Ratio from the image resolution
    Specify the x-, y-, and z-coordinates from the center of the image, using 50 [dip] as the measurement unit.
  - Rotational angle
    Using the device tilt as the y-axis, specify the rotational angle of the x-, y-, and z-coordinates relative to the coordinate system, using the center of the screen as the reference point.

- Tap action
  - URL action
    URL containing the content that will be displayed when the overlay is tapped.
  - Script action
    Script that will be executed when the overlay is tapped. To specify a script for the script action, specify a JavaScript method to be created when the application is mounted.

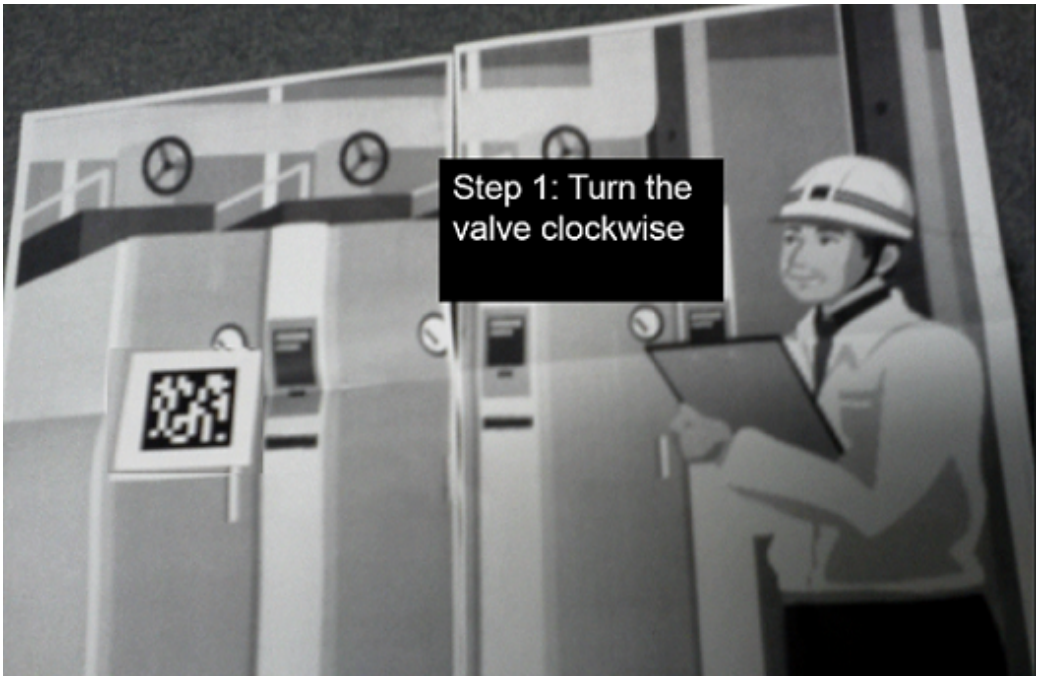
Example

Table 2.3 Example design of AR overlay definition using AR marker

Basic information	AR overlay definition name		Job sample_Valve step
	Recognition mode		AR marker
	AR marker ID		1
Text information	Text		Step 1: Turn valve clockwise
	Font size		20
	Font color		#[B00000] Transparency 255
	Background color		#[FFC0C0] Transparency 255
	Text wrapping		Enabled
	Area height		80
	Area width		160
Position information	Projection		2D
	Translation	x	4.0
		y	1.5
		z	0.0
	Rotation	x	0.0

		y	0.0
		z	0.0
	Scale	x	1.0
		y	1.0
		z	1.0
Other	Use tap action		Yes
	Script action		Apl.shiftScene(true)

Figure 2.1 Example display of AR overlay definition



2.3.6 Designing a user-defined table

Use a user-defined table when using external data in the overlay application. Data that will not be updated, such as fixed external data and external data URLs, can be used in an overlay by specifying it in the AR overlay definition. If data is likely to be updated or you want to accumulate daily data, use a user-defined table.

Design the following items when designing a user-defined table:

- Design items
  - Data table name
    Corresponds to the table name in a database
  - Attribute
    Corresponds to column information in a relational database
    - Attribute name
      Name of the attribute
    - Data type
      Type of the data to be stored in the attribute. Select LONG, FLOAT, or STRING.

Point

- You can retrieve the latest data by creating the registration datetime as an attribute and sorting the data in descending order when retrieving it. You can display overlays using the latest data by including a process in the overlay application for embedding the latest data into the AR overlay definition data.
- The search feature provided by the AR processing server in Interstage AR Processing Server enables you to search for the same attribute name across multiple tables. The user-defined tables in the sample application (AR marker) are designed using the attribute names (such as ar_id, ar_name, and ar_description) common to the product-standard tables (scenario, scene, AR overlay definition). You do not need to use attribute names that are common to all tables. Design the data tables according to your use.
- Note that AR processing servers do not automatically append a value to common attributes including ar_registrationtime and ar_modificationtime even if you specify the same attribute name in the user-defined tables.
- The data type LONG is equivalent to the long type in Java. Depending on the programming language and application environment, the product may not be able to handle any long type data. For example, the number type in JavaScript rounds numerical values of 17 digits or more. Determine the range of values according to the environment you intend to use.

Example

Design a user-defined table (usr_sample) for storing the name of the worker (usr_name) and temperature (usr_temperature) as inspection information in the sample application (AR marker).

Table 2.4 Example of user-defined table

Attribute name	Data type	Remarks
ar_id	LONG	User data ID
ar_name	STRING	User data name
ar_description	STRING	Description or memo
ar_registrationtime	LONG	Registration datetime
ar_modificationtime	LONG	Update datetime
usr_name	STRING	User name
usr_temperature	LONG	Inspection temperature

2.4 Example external specifications

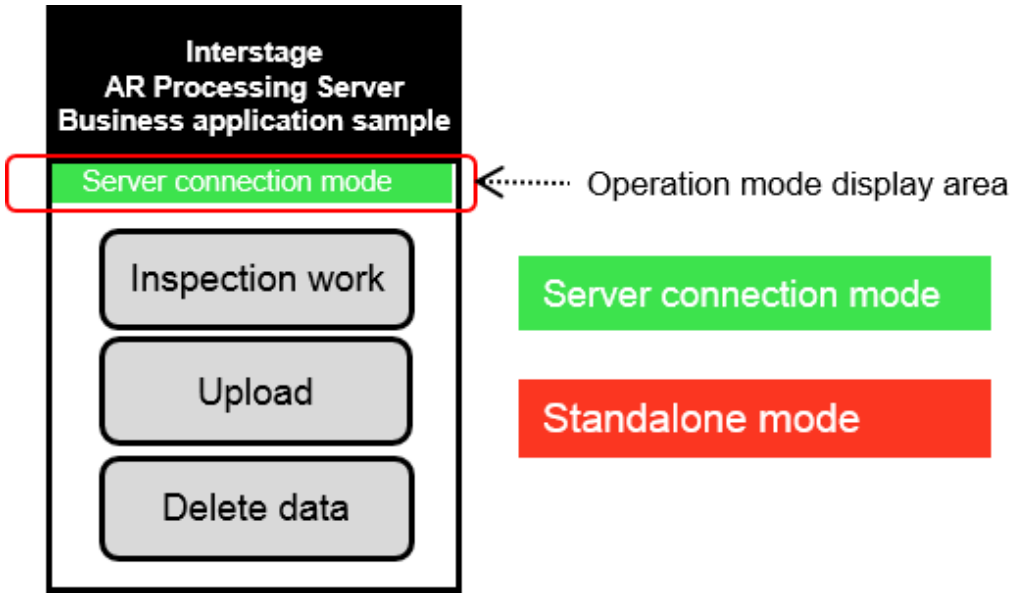
The method of applying external specifications depends on the job content. This section explains external specifications, using the sample application (AR marker) as an example.

Interstage AR Processing Server provides a web application to system users, so determine the following types of content for each screen:

- Processing when a screen is displayed
- Screen GUIs
- Operation content on the screens

2.4.1 index.html

This is the start screen of the sample application (AR marker). The scenario name (in this case, "Inspection work") is not initially displayed in the topmost button - it is only displayed after the scenario is retrieved. You can delete web storage saved to an offline environment and offline storage data. You can upload saved data by clicking **Upload**. This page changes the display of the operation mode display area according to the operation mode.



This page performs the following:

- Stops the camera
- Enables offline storage
- Retrieves data saved to web storage
- Determines the operation mode
  - Changes the display of the display area according to the operation mode
  - In server connection mode: Retrieves the scenario list from the AR processing server and displays the scenario name in the button
  - In standalone mode: Displays the scenario name retrieved from web storage in the button
    If no scenario list is saved to web storage, this page displays a message.

Operation when Inspection work is clicked

Redirects to prework.html

Operation when Upload is clicked

Uploads data according to the operation mode

- In server connection mode: Uploads data to the AR processing server
- In standalone mode: Displays a dialog box prompting the user to run the process in server connection mode

Operation when Delete data is clicked

- Deletes web storage data
- Re-registers the scenario list in web storage
- Deletes offline storage data
- Displays the **Data was deleted** message if data is deleted successfully

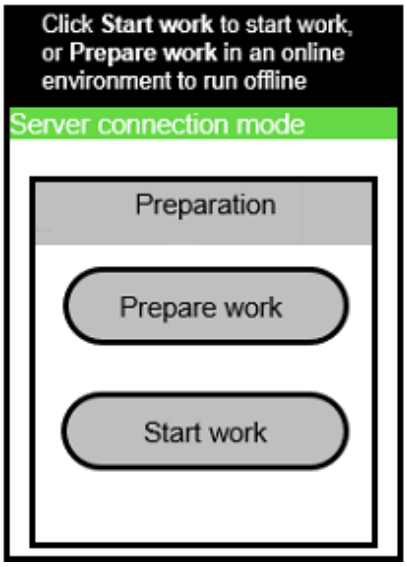
2.4.2 prework.html

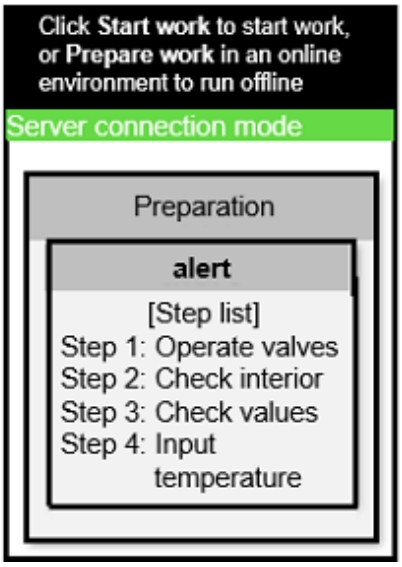
This is the preparation screen for inspection work. You can download user-defined tables, AR overlay definitions, and external resources in preparation for offline operation. For online use, you can start work without downloading anything.

This page performs the following:

- Stops the camera
- Retrieves data saved to web storage
- Determines the operation mode

- Changes the display of the display area according to the operation mode
- In server connection mode: Retrieves the scene list from the AR processing server and displays it as a message
- In standalone mode: Displays the scene list retrieved from web storage as a message. If no scene is saved to web storage, this page displays a message and returns to index.html.





Operation when Prepare work is clicked

- Determines the operation mode
  - In server connection mode:
    - Retrieves the user-defined data and the AR overlay definition data from the AR processing server. It embeds the user-defined data in a specific AR overlay definition and adds it to the renderer. When the AR overlay definition is added to the renderer, external resources are downloaded. This page displays a message when download finishes.
    - The query sent to the executing AR processing server is saved to offline storage and is available during offline use. You can search for all scenes at once, but that means each search retrieves all data from offline storage when you need the data for one scene only. This page therefore runs a separate search for each scene.
- In standalone mode:
  Checks the operation mode and network status, and then prompts the user to run the process again.

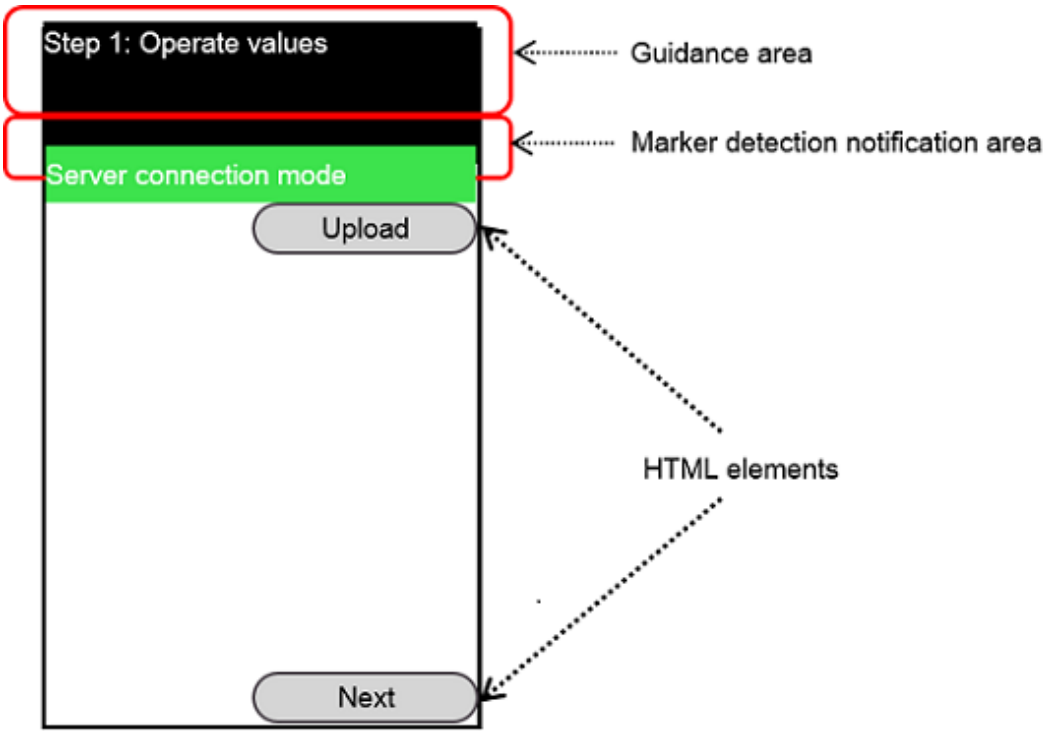
Operation when Start work is clicked

Redirects to work.html

2.4.3 work.html

This is the inspection work screen. This page performs the following:

- Retrieves data saved to web storage
- Changes the HTML according to the operation mode and current scene
- Starts the camera
- Adds the AR marker detection event listener
- Adds the screen tap event listener



Operation when Previous/Next is clicked

- Switches from the current scene to the previous or next scene. **Previous/Next** is not displayed if you are viewing the first or last scene.
- Changes the display of the guidance area at the top of the screen
- Retrieves the AR marker that was recognized
  - If the AR marker ID is 1, retrieves the AR overlay definition according to the operation mode
    - In server connection mode: Retrieves the AR overlay definition for the current scenario ID and scene ID from the AR processing server. If the scene ID is 4, this page also retrieves the user-defined data and embeds it into the AR overlay definition.
    - In standalone mode: Retrieves the AR overlay definition from offline storage and sets it into the renderer. If the scene ID is 4, this page embeds the user-defined data into the AR overlay definition.
- If offline storage is being used, runs the same query as the one downloaded beforehand
- Performs the following when the AR marker is detected:
  - If the AR marker with the AR marker ID of 1 is detected:
    - Displays **Marker 1 was detected** in the AR marker detection notification area
    - Retrieves the AR overlay definition according to the operation mode
    - In server connection mode: Retrieves the AR overlay definition for the current scenario ID and scene ID from the AR processing server. If the current scene ID is 4, this page also retrieves the user-defined data and embeds it into the AR overlay definition.
    - In standalone mode: Retrieves the AR overlay definition from offline storage and sets it into the renderer. If the current scene ID is 4, this page embeds the user-defined data into the AR overlay definition.
  - If the AR marker with the AR marker ID of 1 is lost:
    - Deletes the display of the AR marker detection notification area

Redirects to comment.html

Deletes the AR marker detection event listener simultaneously to the redirection process

Processing when Upload is clicked

- Uploads data according to the operation mode
- In server connection mode: Uploads post data to the AR processing server
- In standalone mode: Displays an alert prompting the user to run the process in server connection mode

2.4.4 comment.html

This is the screen for inputting and registering the inspection value. This page performs the following:

- Stops the camera
- Retrieves data saved to web storage
- Changes the HTML according to the operation mode

Operation when OK is clicked

- Retrieves the name from the form and displays a message if the name has not been entered or contains more than 30 characters
- Retrieves the temperature from the input form and, if the temperature has not been entered or is not a number between 0 and 999, displays a message
- Creates a data query for posting to the AR processing server, and saves the input data and post data to web storage
- Uploads data according to the operation mode
- In server connection mode: Uploads post data to the AR processing server
- In standalone mode: Displays a message prompting the user to run the process in server connection mode
- Redirects to work.html

Operation when Cancel is clicked

Redirects to work.html

2.5 Example internal specifications

The method of applying internal specifications depends on the job content. This section explains internal specifications, using the sample application (AR marker) as an example.

2.5.1 Specifications common to all screens

2.5.1.1 Cache file

This configuration file caches web resources using the features of HTML 5. It caches all the web resources to be used.

```
CACHE MANIFEST

#ver 20130702_0001

#Specify the resources to be cached.
CACHE:
index.html
prework.html
work.html
comment.html
gyoumu.css
apl_sample/apl.js
ar/ar.js

#Specify the resources that must be connected to the server.
NETWORK:
*
```

```
#Specify alternative resources to be used if the preferred resources cannot be accessed.
FALLBACK:
```

2.5.1.2 Properties

Variable	Type	Description	Saved to web storage
Apl.useScenarioId	number	Scenario to be used in the sample application (AR marker). Can be changed.	
Apl.operationMode	string	Contains operation mode.	Y
Apl.scenarioList	Array<>	Contains the scenario list. It is an object that uses the ID as an index, and also contains the scenario name (name) and scenario description (description) as properties.	Y
Apl.scenarioId	number	Contains the current scenario ID.	Y
Apl.sceneList	Array<>	Contains the scene lists. It is an object that uses the ID as an index, and also contains the scene name (name) and scene description (description) as properties.	Y
Apl.sceneId	number	Contains the current scene ID.	Y
Apl.superimposedGraphics	Array<Array<>>	A 3-dimensional array containing scene IDs, AR marker IDs and AR overlay definitions. 1D: Scene ID, 2D: AR marker ID, 3D: AR overlay definition	Y
Apl.postData	Object	Data to be sent to the AR processing server when entering inspection values. query: Query string sent to the AR server body: Data string registered in the AR server	Y
Apl.userData	Object	Contains the latest inspection data. name: User name temperature: Inspection temperature time: Inspection time	Y
Apl.listenerId	string	Listener ID added for marker detection.	
Apl.completionNotice	Object	Used when running multiple downloads, to determine if download has completed.	

2.5.1.3 Classes

Class	Description	Property
Apl.Quad	Class for expressing Quad for the AR processing server.	qtypeName
		id
		qvalues
		version
Apl.QValue	Class for expressing Qvalue for the AR processing server.	qtypeName
		qentityid
		qattributeName
		stringValue
		longValue
		floatValue
		version
Apl.QuadQuery	Class for expressing quad queries for the AR processing server.	type
		limitRange
		qattributeOrderIndexRange
		qtypeNameRange
		whereExpressions
		sortOrderExpressions
Apl.Range	Class for expressing quad query ranges for the AR processing server.	start
		end
Apl.QuadQueryExpression	Class for expressing quad query expressions for the AR processing server.	qattributeNameRanges
		qvalueRanges
		qvalueType
		desc

2.5.1.4 Methods

Method	Outline	Argument	Description
Apl.noop(result)	This method does nothing. Specify it when no operation is to be performed at callback.	_result : AR.Native.Result	Native result object
Apl.log(message, detail)	Creates an error message from a JavaScript error or string and displays an alert. If Detail is specified, it is added and output to the log.	_message: String or Error	Error message text
		_detail: String	Detailed error information used for log output
Apl.getLocalStorageData()	Retrieves all data saved to web storage.	None	
Apl.*** Error(result)	Method for onError for various APIs in the JavaScript library. Displays details of an error as an alert.	_result: AR.Native.Result	Native result object
Apl.makeQuadQuery(qQuery)	Generates a quad search string from the QuadQuery class.	_qQuery: QuadQuery	Apl.QuadQuery object

Method	Outline	Argument	Description
Apl.showOperationMode()	Changes the HTML according to the operation mode.	None	

2.5.1.5 Processes

Web storage

HTML and JavaScript do not save the data held during screen redirection and suspension of an application. All data to be held in common after screen redirection is saved to web storage. Since the timing of screen redirection and suspension is not predictable, web storage saves each data item individually whenever a change is made to it. In addition, the onload function, which is executed for all screens, retrieves data from web storage.

Errors

When an error occurs in JavaScript or in AR native, an error message is displayed as an alert and output to a log.

```
//Outputs error log.
Apl.log = function (_message, _detail) {
  if (_message == null) return;
  var code = 0;
  var message;

  if (_message instanceof Error) { //For an Error object
    if (typeof _message.code == 'number') code = _message.code;
   message = _message.componentName == null ? _message : _message.componentName + " : " + _message;
    if (_message.cause != null) message += " Cause : " + _message.cause;
  } else message = _message; //For a non-Error object

  //Displays message as an alert.
  alert(message);
  //Outputs message + detail to a log.
  if (_detail) message += "\n" + detail;
  try {
    AR.Log.log(AR.Log.LevelType.ERROR, code, message, Apl.noop, Apl.logError);
  } catch (e) {
    alert("Failed to output log."  + e);
  }
};
```

Example

- When an error occurs in JavaScript

```
try {
  AR.Camera.stopCameraView(onSuccess, onError);
} catch (e) {
  Apl.log(e);
}
```

- When an error occurs in AR native

```
onError = function (_result) {
  var message = "Failed to start camera."
 ;
  var detail = _result.getStatus() + "\n" + _result.getValue();
  Apl.log(message, detail);
};
```

2.5.2 index.html

2.5.2.1 Retrieving the application cache

Depending on the network status, retrieval of the application cache may fail, so reload the application cache until you can successfully add the event listener

```
//Confirms retrieval of the application cache
window.applicationCache.addEventListener("cached", function(){
  localStorage.setItem("cacheError","0");
    alert("Successfully retrieved the web cache");
}, false);
window.applicationCache.addEventListener("updateready", function(){
  window.applicationCache.swapCache();
    if(confirm("An update is available for the application. Update now?")) {
      location.reload();
    }
}, false);
window.applicationCache.addEventListener("error", function(){
  if(window.applicationCache.status == window.applicationCache.UNCACHED){
    if(localStorage.getItem("cacheError")) var cacheError =
parseInt(localStorage.getItem("cacheError"),10)+1;
    else var cacheError = 1;
    if(cacheError>=20){ //Regards as a failure if the cache cannot be retrieved within 20 reloads
      alert("Failed to retrieve the web cache. Restart.") ;
      localStorage.setItem("cacheError","0");
    } else {
      localStorage.setItem("cacheError",cacheError);
      location.reload();
    }
  }
  }, false);
```

2.5.2.2 onload processing

After stopping the camera and retrieving data saved to web storage, determine the operation mode

```
Apl.onloadIndex = function(){
  // Stops the camera.
  AR.Camera.stopCameraView(Apl.noop, Apl.stopCameraViewError);
  //Enables offline storage.
  AR.Data.useOfflineStorage(true, Apl.noop,Apl.useOfflineStorageError);
//Gets stored data from the web storage.
  Apl.getLocalStorageData();
  Apl.scenarioId = Apl.useScenarioId;
//Gets the operation mode and saves it to Apl.operationMode.
  AR.OS.getOperationMode(Apl.getOperationModeSuccess ,Apl.getOperationModeError);
};
```

Processing when the operation mode is determined successfully

```
Apl.getOperationModeSuccess = function (_result) {
  Apl.operationMode = _result.getValue();
  localStorage.setItem("operationMode", Apl.operationMode);
  //Changes the HTML display according to the operation mode.
  Apl.showOperationMode();
  if (Apl.operationMode == "serverMode") { //For server connection mode
    //Gets scenario lists from the AR processing server.

    Apl.getScenario();
  } else { //For standalone mode
    //Displays the scenario button.
```

```
    Apl.showScenarioButton();
  }
};
```

Generate a query for retrieving the scenario and retrieve the scenario list from the AR processing server

```
Apl.getScenario = function () {
  //Creates query objects for quads searches.
  var query = new Apl.QuadQuery();
  query.type = "RECORDSANDCOUNT";
  query.limitRange = new Apl.Range(1, 10);
  query.qattributeOrderIndexRange = new Apl.Range(1, 10);
  //Specifies the qtype to which scenarios are registered.
  query.qtypeNameRanges = new Apl.Range("arscn_scenario");
  //Specifies scenario IDs.
  query.whereExpressions = new Apl.QuadQueryExpression(new Apl.Range("ar_id"), new
Apl.Range(Apl.useScenarioId), "LONG");

  //Converts to a string.
  query = Apl.makeQuadQuery(query);

  //Gets scenario lists from the AR processing server.
  AR.Data.getArServerData(query, true, Apl.getScenarioSuccess, Apl.getArServerDataError);
};
```

Processing when the scenario is retrieved successfully from the AR processing server

```
Apl.getScenarioSuccess = function (_result) {

  //Initializes
  Apl.scenarioList = null;
  //Extracts scenarios from the results and stores them in Apl.scenarioList.
  Apl.extractScenario(_result.getValue());
  //Stores scenarioID in web storage.
  localStorage.setItem("scenarioId", Apl.scenarioId);
  //Stores registered scenarios in web storage.
  if (Apl.scenarioList != null)
    localStorage.setItem("scenarioList", JSON.stringify(Apl.scenarioList));

  //Displays a button from the retrieved scenario.
  Apl.showScenarioButton();
};
```

Extract scenarios from the JSON object

```
Apl.extractScenario = function (_data) {
  //Number of retrieved records.
  var recordLength = _data.records.length;

  for (var recordIndex = 0; recordIndex < recordLength; recordIndex++) {
    //Looks up records one by one.
    var record = _data.records[recordIndex];
    var valueLength = record.qvalues.length;
    var scenarioId;
    var value = new Object();

    //Gets only the qvalues that will be used (determined by qattributeName)
    for (var valueIndex = 0; valueIndex < valueLength; valueIndex++) {
      var qvalue = record.qvalues[valueIndex];
      if (qvalue.qattributeName == "ar_name") { //Scenario name
        value.name = qvalue.stringValue;
      }
      if (qvalue.qattributeName == "ar_description") { //Scenario description
        value.description = qvalue.stringValue;
```

```
      }
      if (qvalue.qattributeName == "ar_id") { //Scenario ID
        scenarioId = qvalue.longValue;
      }
    }
    if (value.name != null) {
      //Stores in Apl.scenarioList.
      if (Apl.scenarioList == null) Apl.scenarioList = new Array();
      Apl.scenarioList[Apl.scenarioId] = value;
    }
  }
};
```

Print the retrieved scenario name onto the button

```
Apl.showScenarioButton = function () {
  if (Apl.scenarioList != null && Apl.scenarioList[Apl.scenarioId] != null) {
    document.getElementById("scenario_select").value = Apl.scenarioList[Apl.scenarioId].name;
    document.getElementById("scenario_select").style.visibility = "visible";
    document.getElementById("scenario_select").onclick = Apl.changePreWork;
  } else {
    alert("Could not retrieve a scenario. Check the operation mode and network status, and then
retry the process.") ;
  }
};
```

2.5.2.3 Processing when Inspection work is operated

Redirect to prework.html

```
Apl.changePreWork = function () {
  //Redirects to prework.html.
  location.href = "prework.html";
};
```

2.5.2.4 Processing when Delete data is operated

Delete data from web storage and offline storage. The scenario and operation mode are retrieved by index.html, so save the scenario list and operation mode to web storage again

```
Apl.clearCache = function () {
  //Deletes all data from web storage.
  localStorage.clear();
  if (Apl.scenarioList != null)
  //Saves scenarios to web storage again.
    localStorage.setItem("scenarioList", JSON.stringify(Apl.scenarioList));
  if (Apl.operationMode != null)
  //Saves operation modes to web storage again.
    localStorage.setItem("operationMode", JSON.stringify(Apl.operationMode));
  //Deletes data from offline storage.
  AR.Data.clearResourceStorage(Apl.clearStorageSuccess, Apl.clearResourceStorageError);
};

//Callback function used when offline storage is deleted successfully.
Apl.clearStorageSuccess = function (_result) {
  alert("The data was deleted.") ;
};
```

2.5.3 prework.html

2.5.3.1 onload processing

Retrieve data saved to web storage, change the HTML according to the operation mode, and then retrieve the scene list

```
Apl.onloadPreWork = function () {
  // Stops the camera.
  AR.Camera.stopCameraView(Apl.noop, Apl.stopCameraViewError);
  //Gets data from web storage.
  Apl.getLocalStorageData();
  //Changes the HTML display according to the operation mode.
  Apl.showOperationMode();
  //Gets scenes according to the operation mode.
  if (Apl.operationMode == "serverMode") { //For server connection mode
    //Gets a scene list for the current scenario ID and stores it in Apl.sceneList.
    Apl.getScene(Apl.scenarioId);
  } else { //For standalone mode
    //Displays a scene list retained in Apl.sceneList.
    Apl.showSceneList();
  }
};
```

Generate a query for retrieving scenes and retrieve data from the AR processing server

```
A Apl.getScene = function (_scenarioId) {
  //Creates query objects.
  var query = new Apl.QuadQuery();
  query.type = "RECORDSANDCOUNT";
  query.limitRange = new Apl.Range(1, 10);
  query.qattributeOrderIndexRange = new Apl.Range(1, 10);
  //Specifies the qtype to which a scene is registered.
  query.qtypeNameRanges = new Apl.Range("arsen_scene");
  //Specifies the scene id range.
  query.whereExpressions = new Array(new Apl.QuadQueryExpression(new Apl.Range("ar_id"), new
Apl.Range(1, 4), "LONG"));
  if (typeof _scenarioId != 'undefined')
    //Specifies the scenario ID.
   query.whereExpressions.push(new Apl.QuadQueryExpression(new Apl.Range("arscn_scenarioid"), new
Apl.Range(_scenarioId), "LONG"));

  //Converts to a string.
  query = Apl.makeQuadQuery(query);

  //Gets scenes from the AR processing server.
  AR.Data.getArServerData(query, true, Apl.getSceneSuccess, Apl.getArServerDataError);
};
```

Processing when scenes are retrieved successfully from the AR processing server

```
Apl.getSceneSuccess = function (_result) {
  //Initializes
  Apl.sceneList = null;
  //Extracts scenes from the results and stores them in Apl.scenarioList.
  Apl.extractScene(_result.getValue());
  //Stores registered scenes in web storage.
  if (Apl.sceneList != null)
    localStorage.setItem("sceneList", JSON.stringify(Apl.sceneList));
  //Displays a scene list.
  Apl.showSceneList();
};
```

Extract scenes

```
Apl.extractScene = function (_data) {
  //Number of retrieved records.
  var recordLength = _data.records.length;
  for (var recordIndex = 0; recordIndex < recordLength; recordIndex++) {
    //Looks up records one by one.
    var record = _data.records[recordIndex];
    var valueLength = record.qvalues.length;
    var value = new Object();
    var sceneId;
    //Gets only the qvalues that will be used (determined by qattributeName)
    for (var valueIndex = 0; valueIndex < valueLength; valueIndex++) {
      var qvalue = record.qvalues[valueIndex];
      if (qvalue.qattributeName == "ar_name") { //Scene name
        value.name = qvalue.stringValue;
      }
      if (qvalue.qattributeName == "ar_description") { //Scene description
        value.description = qvalue.stringValue;
      }
      if (qvalue.qattributeName == "ar_id") { //Scene ID
        sceneId = qvalue.longValue
      }
    }
    if (value.name != null && sceneId != null) {
      //Stores in Apl.sceneList.
      if (Apl.sceneList == null) Apl.sceneList = new Array();
      Apl.sceneList[sceneId] = value;
    }
  }
};
```

Display the scene list

```
Apl.showSceneList = function () {
  if (Apl.sceneList != null) {
    var show = "[List of steps]\n";
    for (scene in Apl.sceneList) {
      if (Apl.sceneList[scene] != null)
        show += "step" + scene + " : " + Apl.sceneList[scene].name + "\n";
    }
    alert(show);
  } else {
    alert("Could not get scene. Check the operation mode and network status, and then retry the
process.") ;
    location.href = "index.html";
  }
};
```

2.5.3.2 Operation for Prepare work

Download resources from AR processing server

```
Apl.resourceDownload = function(){
  if(Apl.operationMode == "serverMode"){ //For server connection mode
    //Sets download notifications.
    Apl.completionNotice.notice = true;
    Apl.completionNotice.count++;

    //Gets user-defined data and stores it in Apl.userData.
    Apl.getUserData(true);

    //Gets AR overlay definition data for each scene and stores it in Apl.superimposedGraphics.
```

```
    //Queries and responses are saved to offline storage.
    for(scene in Apl.sceneList){
      Apl.completionNotice.count++;
      Apl.getSuperimposedGraphicData(true, new Apl.Range(Apl.scenarioId), new
Apl.Range(parseInt(scene)),new Apl.Range(1));
    }
  } else { //For standalone mode
    alert("Run in server connection mode.") ;
  }
}
```

Generate a query for retrieving a user-defined data and retrieve data from the AR processing server

```
Apl.getUserData = function (_isSuperReload) {
  //Creates query objects.
  var query = new Apl.QuadQuery();
  query.type = "RECORDSANDCOUNT";
  query.limitRange = new Apl.Range(1, 1);
  query.qattributeOrderIndexRange = new Apl.Range(1, 10);
  //Specifies the qtype to which user-defined data is registered.
  query.qtypeNameRanges = new Array(new Apl.Range("usr_sample"));
  //Specifies ar_id.
  query.whereExpressions = new Array(new Apl.QuadQueryExpression(new Apl.Range("ar_id"), new
Apl.Range(1), "LONG"));
  //Sorts the registration time (ar_registrationtime) in descending order and gets the latest data.
  query.sortOrderExpressions = new Array(new Apl.QuadQueryExpression(new
Apl.Range("ar_registrationtime"), new Apl.Range(0, 9000000000000), "LONG", true));
  //Converts to a string.
  query = Apl.makeQuadQuery(query);

  //Gets user-defined data from the AR processing server.
  AR.Data.getArServerData(query, _isSuperReload, Apl.getUserDataSuccess,
Apl.getArServerDataError);
};
```

Processing when the user-defined data is retrieved successfully from the AR processing server

```
Apl.getUserDataSuccess = function(_result){
  //Extracts the required data from the results and stores it in Apl.userData.
  Apl.extractUserData(_result.getValue());
  //Stores registered user-defined data in web storage.
  if(Apl.userData != null)
    localStorage.setItem("userData", JSON.stringify(Apl.userData));

  //Determines download completion notifications.
  Apl.checkCompletion();
};
```

Extract the user-defined data from the JSON object

```
Apl.extractUserData = function (_data) {
  //Number of retrieved records.
  var recordCount = _data.records.length;
  for (var recordIndex = 0; recordIndex < recordCount; recordIndex++) {
    //Looks up records one by one.
    var record = _data.records[recordIndex];
    var valueLength = record.qvalues.length;
    var value = new Object();
    //Gets only the qvalues that will be used (determined by qattributeName)
    for (var valueIndex = 0; valueIndex < valueLength; valueIndex++) {
      var qvalue = record.qvalues[valueIndex];
      if (qvalue.qattributeName == "usr_name") { //User name
        value.name = qvalue.stringValue;
      }
```

```
      if (qvalue.qattributeName == "usr_temperature") { //Inspection temperature
        value.temperature = qvalue.longValue;
      }
      if (qvalue.qattributeName == "ar_registrationtime") { //Registration time
        value.time = qvalue.longValue;
      }
    }
    //Compares the registration times for the retrieved data and stores the latest data in
Apl.userData.
    if (value.name != null)
      if (Apl.userData == null || Apl.userData.time < value.time) {
        Apl.userData = value;
      }
  }
}
```

Generate a query for retrieving the AR overlay definition and retrieve data from AR processing server

```
Apl.getSuperimposedGraphicData = function(_isSuperReload, _scenarioId, _sceneId, _markerId){
  //Creates quad query objects.
  var query = new Apl.QuadQuery();
  query.type = "RECORDSANDCOUNT";
  query.limitRange = new Apl.Range(1,10);
  query.qattributeOrderIndexRange = new Apl.Range(1,10);
  //Specifies the qtype to which AR overlay definition data is registered.
  query.qtypeNameRanges = new Array(new Apl.Range("arpoiarmk_default"));
  query.whereExpressions = new Array();

  //Specifies the scenario ID.
  if(typeof _scenarioId != 'undefined'){
    if(typeof _scenarioId == 'number' ) _scenarioId = new Apl.Range(_scenarioId);
    query.whereExpressions.push(new Apl.QuadQueryExpression(new Apl.Range("arscn_scenarioid"),
_scenarioId, "LONG"));
  }
  //Specifies the scene ID.
  if(typeof _sceneId != 'undefined'){
    if(typeof _sceneId == 'number' ) _sceneId = new Apl.Range(_sceneId);
    query.whereExpressions.push(new Apl.QuadQueryExpression(new Apl.Range("arsen_sceneid"),
_sceneId, "LONG"));
  }
  //Specifies the marker ID.
  if(typeof _markerId != 'undefined'){
    if(typeof _markerId == 'number' ) _markerId = new Apl.Range(_markerId);
    query.whereExpressions.push(new Apl.QuadQueryExpression(new Apl.Range("armk_markerid"),
_markerId, "LONG"));
  }
  //Converts to a string.
  query = Apl.makeQuadQuery(query);

  //Gets AR overlay definition data from the AR processing server.
    AR.Data.getArServerData(query, _isSuperReload, Apl.getSuperimposedGraphicDataSuccess,
Apl.getArServerDataError);
};
```

Processing when the AR overlay definition data is retrieved successfully from the AR processing server

```
Apl.getSuperimposedGraphicDataSuccess = function(_result){
  //Extracts AR overlay definition data from the results.
  var contents = Apl.extractSuperimposedGraphic(_result.getValue());

  //Stores the extracted AR overlay definition data by scene ID and marker ID in
Apl.superimposedGraphics.
  for(scene in contents){
    if(Apl.superimposedGraphics == null) Apl.superimposedGraphics = new Object();
```

```
    if(Apl.superimposedGraphics[scene] == null) {
      Apl.superimposedGraphics[scene] = contents[scene];
      //Sets the AR overlay definition data in the native AR display layer.
      Apl.setARContents(parseInt(scene));
    } else {
      for(marker in contents[scene]){
        Apl.superimposedGraphics[scene][marker] = contents[scene][marker];
        //Sets the AR overlay definition data in the native AR display layer.
        Apl.setARContents(parseInt(scene),parseInt(marker));

      }
    }
  }
  //If AR overlay definition data exists, saves it to web storage.
  if(Apl.superimposedGraphics != null){
    localStorage.setItem("ARContents", JSON.stringify(Apl.superimposedGraphics));
  }
  //Verifies download completion notifications.
  Apl.checkCompletion();
};
```

Extract the AR overlay definition data from the JSON object

```
Apl.extractSuperimposedGraphic = function(_data){
  //Number of retrieved records.
  var recordCount = _data.records.length;
  var superimposedGraphic = new Object();
    for(var recordIndex = 0; recordIndex < recordCount; recordIndex++){
      //Looks up records one by one.
      var record = _data.records[recordIndex];
      var valueLength = record.qvalues.length;
      var arName, sceneId, markerId;
      var value;
      //Gets only the qvalues that will be used (determined by qattributeName)
      for(var valueIndex = 0; valueIndex < valueLength; valueIndex++){
        var qvalue = record.qvalues[valueIndex];
        if(qvalue.qattributeName == "ar_name"){ //AR overlay definition name
          arName = qvalue.stringValue;
        }
        if(qvalue.qattributeName == "arsen_sceneid"){ //Scene ID
          sceneId = qvalue.longValue;
        }
        if(qvalue.qattributeName == "armk_markerid"){ //Marker ID
          markerId = qvalue.longValue;
        }
        if(qvalue.qattributeName == "arpoi_superimposedgraphic"){ //AR overlay definition
          value = AR.Renderer.parseSuperimposedGraphic(qvalue.stringValue);
        }
      }
     //Ensures that AR overlay definition data name, Scene ID, Marker ID and AR overlay definition
data are not null.
      //Scene ID, Marker ID and AR overlay definition data are required for overlays.
     //AR overlay definition data name is checked in order to identify the overlay when embedding
user-defined data.
      if(arName != null && sceneId!=null && markerId != null && value != null){
        //Embeds the user-defined data in the AR overlay definition.
        value = Apl.insertUserData(value, arName);
        if(superimposedGraphic[sceneId] == null) superimposedGraphic[sceneId] = new Object();
        if(superimposedGraphic[sceneId][markerId] == null) superimposedGraphic[sceneId][markerId]
= new Array(value);
        else superimposedGraphic[sceneId][markerId].push(value);
      }
```

```
    }
  return superimposedGraphic;
};
```

Perform the process for embedding the user-defined data when ar_id of the AR overlay definition data is the specified value

```
Apl.insertUserData = function (_superimposedGraphic, _arName) {
  if (_arName == "Job sample_inspection value" && Apl.userData != null) {
    //Creates strings for embedding.
    var registrationDate = new Date(Apl.userData.time);
    var month = registrationDate.getMonth() + 1;
    var vDate = month + "/" + registrationDate.getDate();
    var commentStr = "[" + vDate + "  " + Apl.userData.name + "]The temperature is " +
Apl.userData.temperature + " degrees C";

    //superimposedGraphic -> SquareModelGraphic -> Edits the text for TextTexture.
    _superimposedGraphic.getGraphic().getTexture().setText(commentStr);
  }
  return _superimposedGraphic;
};
```

Set the AR overlay definition data with the specified scene ID and AR marker ID in the native AR display layer

```
Apl.setARContents = function(_sceneId, _markerId){
  if(Apl.superimposedGraphics[_sceneId] != null){ //If there is AR overlay definition data for the
specified scene ID
    if(_markerId == null){ //If no marker ID was specified, sets the AR overlay definition data
for all scenes.
      for(marker in Apl.superimposedGraphics[_sceneId]){
        //Creates a coordinate system for AR markers and specifies marker IDs.
        var coordinateSystem = new AR.Renderer.FJARMarkerCoordinateSystem();
        coordinateSystem.setValue(parseInt(marker));
        //Deletes AR overlay content for applicable marker IDs.
        AR.Renderer.remove(coordinateSystem, Apl.noop, Apl.removeError);
        //Sets the AR overlay definition data in the native AR display layer.
        AR.Renderer.put(coordinateSystem, Apl.superimposedGraphics[_sceneId][parseInt(marker)],
Apl.noop, Apl.putError);
      }
    } else if(Apl.superimposedGraphics[_sceneId][_markerId] != null) { //If AR overlay definition
data exists for the specified marker ID
      //Creates a coordinate system for AR markers and specifies marker IDs.
      var coordinateSystem = new AR.Renderer.FJARMarkerCoordinateSystem();
      coordinateSystem.setValue(_markerId);
      //Deletes AR overlay content for applicable marker IDs
      AR.Renderer.remove(coordinateSystem, Apl.noop, Apl.removeError);
      //Sets the AR overlay definition data in the native AR display layer.
      AR.Renderer.put(coordinateSystem, Apl.superimposedGraphics[_sceneId][_markerId], Apl.noop,
Apl.putError);
    }
  }
};
```

Check if downloading has finished

```
Apl.checkCompletion = function(){

  if(Apl.completionNotice.notice == true){
    Apl.completionNotice.count--;
    if(Apl.completionNotice.count == 0){
      Apl.completionNotice.notice = false;
      alert("The AR overlay definition data was downloaded");
    }
```

```
  }
};
```

2.5.3.3 Operation for Start work

Clear the AR overlay definition registered in the native AR display layer and redirect to work.html

```
Apl.start = function(){
  //Clears all AR overlay definition data set in the native AR display layer.
  AR.Renderer.clear(Apl.clearSuccess,Apl.clearError);
};

//When AR overlay definition data is successfully cleared
Apl.clearSuccess = function(_result){
  //Redirects to work.html.
  location.href="work.html";
};
```

2.5.4 work.html

2.5.4.1 onload processing

Retrieve data saved to web storage, change the screen display, add the screen tap event listener, start the camera, add the event listener when the AR marker is detected, and retrieve data from the AR processing server

```
Apl.onloadWork = function(){
  //Gets all data saved to web storage.
     Apl.getLocalStorageData();
  //Changes the HTML display according to the operation mode.
     Apl.showOperationMode();
  //Changes the HTML display according to the scene.
     Apl.changeSceneView();
  //Adds event listeners for screen taps.
     if(window.navigator.userAgent.match(/(iPad|iPhone|iPod)/i)) //In iOS
          document.addEventListener("touchstart", Apl.clickEvent, false);
     else document.addEventListener("click", Apl.clickEvent, false); //In android and Windows
     //Starts the camera.
     AR.Camera.startCameraView(Apl.noop, Apl.startCameraViewError);
  //Adds event listeners for marker detection.
     AR.Camera.addMarkerListener( Apl.addMarkerListenerSuccess, Apl.addMarkerListenerError,
Apl.onDetectMarker,Apl.noop);

     //Determines the operation mode.
     if(Apl.operationMode == "serverMode"){ //For server connection mode
         //Performs forced reading from the AR Processing Server.
         //If the scene ID is 4, gets the user-defined data.
         if(Apl.sceneId == 4) Apl.getUserData(true);
              Apl.getSuperimposedGraphicData(true, new Apl.Range(Apl.scenarioId), new
Apl.Range(Apl.sceneId), new Apl.Range(1));
     } else { //For standalone mode
          //Gets data from offline storage.
          if(Apl.sceneId == 4) Apl.getUserData(false);
          Apl.getSuperimposedGraphicData(false, new Apl.Range(Apl.scenarioId), new
Apl.Range(Apl.sceneId), new Apl.Range(1));
     }
};
```

2.5.4.2 Operation when an AR marker is detected

If an AR marker with an AR marker ID of 1 is detected, change the display at the top part of the screen

```
Apl.onDetectMarker = function(_result){
  //Includes information about markers detected in an argument.
  var markval = _result.getValue();

  if(markval.markerId == 1){ //If marker 1 is detected
    if(markval.status == true){ //If detected
      //Changes the display of the marker detection notification area.
      document.getElementById("detect").innerHTML ="marker" +markval.markerId+"was detected." ;
      //Determines the operation mode.
      if(Apl.operationMode == "serverMode"){ //For server connection mode
        //Performs forced reading from the AR Processing Server.
        //If the scene ID is 4, gets the user-defined data.
        if(Apl.sceneId == 4) Apl.getUserData(true);
        Apl.getSuperimposedGraphicData(true, new Apl.Range(Apl.scenarioId), new
Apl.Range(Apl.sceneId));
      } else { //For standalone mode
        //Gets data from offline storage.
        if(Apl.sceneId == 4) Apl.getUserData(false);
        Apl.getSuperimposedGraphicData(false, new Apl.Range(Apl.scenarioId), new
Apl.Range(Apl.sceneId));
      }

    } else if(markval.status == false){ //If missing
      //Changes the display of the marker detection notification area.
      document.getElementById("detect").innerHTML ="";
    }
  }
};
```

2.5.4.3 Operation for Previous/Next buttons

Change to the previous/next scene ID, change the screen display, and retrieve data from the AR processing server

```
    Apl.shiftScene = function(_shift){
         if(_shift == true) Apl.sceneId++; //Moves to the next scene.
         else Apl.sceneId--; //Moves to the previous scene.
         //Changes the HTML display according to the scene.
         Apl.changeSceneView();
  //Saves the current scene ID to web storage.
         localStorage.setItem("sceneId", Apl.sceneId);
  //Gets the detected markers.
         AR.Camera.getCurrentMarkers(Apl.getMarkersSuccess, Apl.getCurrentMarkersError);
          //Determines the operation mode.
         if(Apl.operationMode == "serverMode"){ //For server connection mode
             //Performs forced reading from the AR Processing Server.
             //If the scene ID is 4, gets the user-defined data.
             if(Apl.sceneId == 4) Apl.getUserData(true);
              Apl.getSuperimposedGraphicData(true, new Apl.Range(Apl.scenarioId), new
Apl.Range(Apl.sceneId), new Apl.Range(1));
         } else { //For standalone mode
              //Gets data from offline storage.
              if(Apl.sceneId == 4) Apl.getUserData(false);
              Apl.getSuperimposedGraphicData(false, new Apl.Range(Apl.scenarioId), new
Apl.Range(Apl.sceneId), new Apl.Range(1));
         }
    };
```

2.5.4.4 Processing when Upload is operated

If registration data was saved, confirm in the dialog box whether to upload it, and check the network status. If data was not saved, display a message.

```
Apl.onClickUpload = function(){
  if(Apl.postData != null){ //If unregistered data exists
    var result = confirm("Upload?") ;
    if(result){
      if(Apl.operationMode == "serverMode"){ //For server connection mode
        Uploads the saved data.
        Apl.upload();
      } else { //For standalone mode
        alert("The data was saved. Please upload in server connection mode.") ;
        if(location.href != "work.html") location.href = "work.html";
      }
    }
  } else {
    alert("There is no saved data.")
  }
};
```

Upload the saved data

```
Apl.upload = function(){
  //Uploads to the AR processing server.
  AR.Data.postArServerData(Apl.postData.query, Apl.postData.body, Apl.uploadSuccess,
  Apl.postArServerDataError);
};
```

Processing when upload is successful

```
Apl.uploadSuccess = function(_result){
  //Deletes data saved to web storage.
  localStorage.removeItem("postData");
  Apl.postData = null;
  alert("Data was uploaded.") ;
  //Redirects to work.html.
  location.href = "work.html";
};
```

2.5.4.5 Operation when the screen is tapped

Notification to the click event AR native in the HTML layer

```
Apl.clickEvent = function(){
  //Notifies the native application of the click coordinates.
  AR.OS.onBodyClick(event, Apl.noop, Apl.onBodyClickError);
};
```

Processing when the "enter inspection value" icon is tapped

```
Apl.changeCommentpage = function(){
  //Confirms if there is any saved data.
  if(Apl.postData){
    var result = confirm("There is saved data. Discard data and reenter?") ;
    if(result == false) return;
  }
  //Redirects to the page used for entering the inspection value.
  location.href="comment.html";
};
```

Point

When the step overlay is tapped, Apl.shiftScene(true) is called (same processing as when **Previous**/**Next** is clicked).

2.5.4.6 onunload processing

Delete the event listener used for detecting an AR marker

```
Apl.onUnloadWork = function(){
  //Deletes an event listener.
  AR.Camera.removeMarkerListener(Apl.listenerId, Apl.removeListenerSuccess,
Apl.removeListenerError);
};

//Called when deletion of the marker detection event listener was successful.
Apl.removeListenerSuccess= function(_result){
  Apl.listenerId = "";
};
```

Note

Note that onunload events are not generated in the following operating systems:

- Android 4.4 or later
- Windows

2.5.5 comment.html

2.5.5.1 onload processing

```
Apl.onloadComment = function(){
  //Gets all data saved to web storage.
  Apl.getLocalStorageData();
  //Changes the HTML display according to the operation mode.
  Apl.showOperationMode();
  //Stops the camera.
  AR.Camera.stopCameraView(Apl.noop, Apl.stopCameraViewError);
};
```

2.5.5.2 Operation for OK

Retrieve data from the input form

If there is no data, then displays an alert, otherwise generates an object to be posted to the AR processing server and saves it to web storage.

```
Apl.onClickOK = function(){
  //Gets input data.
  var data = Apl.getCommentData();
  if(data){

    Apl.userData = data;
    //Creates query data to be registered in the AR processing server.
    Apl.postData = new Object();
    Apl.postData.query = "quads";
    Apl.postData.body = Apl.createCommentDataQuad(data);

    //Saves to web storage.
```

```
    localStorage.setItem("userData", JSON.stringify(Apl.userData));
    localStorage.setItem("postData", JSON.stringify(Apl.postData));

    if(Apl.operationMode == "serverMode"){ //For server connection mode
      Apl.upload();
    } else { //For standalone mode
      alert("The data was saved. Please upload in server connection mode.") ;
      if(location.href != "work.html") location.href = "work.html";
    }
  }
};
```

Retrieve data from the input form

```
Apl.getCommentData = function(){

  var fName = document.comment_form.f_name.value;
  //Displays an alert if a name has not been entered.
  if(fName == null || fName == ""){
    alert("A name has not been entered. Please enter a name.") ;
    return;
  }
  //Displays an alert if the name is too long.
  if(fName.length>30){
    alert("The name is too long. Specify a name up to 30 characters long.") ;
    return;
  }
  var fTemperature = document.comment_form.f_comment.value;
  //Displays an alert if a temperature has not been entered.
  if(fTemperature == null || fTemperature == ""){
    alert("A temperature has not been entered. Please enter a temperature.") ;
    return;
  }
  //If the temperature length exceeds 3 digits, displays an alert indicating that the temperature
should be between 0 and 999.
  if(fTemperature.length > 3){
    alert("Specify a temperature between 0 and 999.") ;
    return;
  }
  //Creates a user data object.
  var data = new Object();
  data.name = fName;
  data.temperature = parseInt(fTemperature);
  data.time = new Date().getTime();
  return data;
};
```

Create quad data for registration

```
Apl.createCommentDataQuad = function(_data){
  //Creates a QUAD.
  var quad = new Apl.Quad();

  //Sets the QUAD type name.
  quad.qtypeName = "usr_sample";

  //Creates the attribute values for a QUAD.
  //Creates QValue ar_id.
  var arId = new Apl.QValue();
  arId.qtypeName = "usr_sample";
  arId.qattributeName = "ar_id";
  arId.stringValue = null;
  arId.longValue = 1;
  arId.floatValue = null;
```

```
  //Creates QValue ar_name.
  var arName = new Apl.QValue();
  arName.qtypeName = "usr_sample";
  arName.qattributeName = "ar_name";
  arName.stringValue = "inspectionResult";
  arName.longValue = null;
  arName.floatValue = null;

  //Creates QValue ar_description.
  var arDescription = new Apl.QValue();
  arDescription.qtypeName = "usr_sample";
  arDescription.qattributeName = "ar_description";
  arDescription.stringValue = "Inspection result of the AR business sample application" ;
  arDescription.longValue = null;
  arDescription.floatValue = null;

  //Creates QValue ar_registrationtime.
  var arRegistrationtime = new Apl.QValue();
  arRegistrationtime.qtypeName = "usr_sample";
  arRegistrationtime.qattributeName = "ar_registrationtime";
  arRegistrationtime.stringValue = null;
  arRegistrationtime.longValue = _data.time;
  arRegistrationtime.floatValue = null

  //Creates QValue ar_modificationtime.
  var arModificationtime = new Apl.QValue();
  arModificationtime.qtypeName = "usr_sample";
  arModificationtime.qattributeName = "ar_modificationtime";
  arModificationtime.stringValue = null;
  arModificationtime.longValue = _data.time;
  arModificationtime.floatValue = null;

  //Creates QValue usr_name.
  var usrName = new Apl.QValue();
  usrName.qtypeName = "usr_sample";
  usrName.qattributeName = "usr_name";
  usrName.stringValue = _data.name;
  usrName.longValue = null;
  usrName.floatValue = null;

  //Creates QValue usr_temperature.
  var usrTemperature = new Apl.QValue();
  usrTemperature.qtypeName = "usr_sample";
  usrTemperature.qattributeName = "usr_temperature";
  usrTemperature.stringValue = null;
  usrTemperature.longValue = _data.temperature;
  usrTemperature.floatValue = null;

  //Sets a QValue for a QUAD.
  quad.qvalues = [arId, arName, arDescription, arRegistrationtime, arModificationtime, usrName,
  usrTemperature]

  //Converts to a string.
  var rtndata = JSON.stringify(quad);
  return rtndata;
};
```

2.6 Creating the system

Create applications and resources to be used in applications based on the designed content.

- Developing applications
  Interstage AR processing server requires the development of an overlay application (web application) for displaying overlays and an overlay application (native application) to be run on smart devices.
  To integrate with an existing system, you must also develop a system integration application.
  Development of each of these applications is explained in "Chapter 4 Developing the overlay application (web application)", "Chapter 5 Developing the overlay application (native application)", and "Chapter 6 Developing the system integration application" respectively.
- Developing resources
  If you need to create resources to be used in AR overlay content and tap actions, develop them using an image editor or similar software.

Chapter 3 Creating 3D models

Interstage AR Processing Server can display 3D models using a smart device. This chapter provides an overview of the supported 3D models and notes on creation.

Note

3D display is not supported if the recognition mode is "Beacon".

3.1 Overview of supported 3D models

Interstage AR Processing Server supports 3D models as AR overlay content, to render three-dimensional objects on the screen. The 3D models supported by Interstage AR Processing Server are made up of polygon information (obj file) and material information (mtl file and texture image in an image file).

- Polygon information
  - obj file
    Contains polygon information about the 3D model (such as the shape and normal vector information).
- Material information
  - mtl file
    Contain material information about the 3D model (such as color, glossiness, and texture).
  - Texture image (image file)
    Create an image file when applying a texture image to a 3D model.

Information

The .obj and .mtl 3D model formats were developed by Wavefront Technologies.

3.1.1 Parameters

The parameters available for obj and mtl files are as follows:

Table 3.1 List of parameters available for obj files

Key	Type	Description
v	Zenith information	Zenith of the polygon
vt	Zenith information	Texture coordinates
vn	Zenith information	Normal vector of the zenith
f	Element information	Surface of the shape
usemtl	Material information	Material file
mtllib	Material information	Material library (*1)

Table 3.2 List of parameters available for mtl files

Key	Type	Description
newmtl	Base material information	Material definition
Ka	Base material information	Material shadow
Kd	Base material information	Material body color
Ks	Base material information	Highlight color
Ns	Base material information	Highlight coefficient
d/Tr	Base material information	Transparency

Key	Type	Description
map_Ka	Base material information	Texture file path for expressing ambient reflection (*2)
map_Kd	Base material information	Texture file path for expressing diffuse reflection (*2)

*1: Only one mtl file can be set per 3D model.

*2: Only one texture file can be set per 3D model. If more than one texture file is set, the one if the first analyzed key in the mtl file will be used.

3.1.2 Recommended size

This section describes the recommended file size for 3D models. For 3D models smaller than the recommended file size, each file is analyzed for about 3 seconds before being displayed as augmented reality (processing time may vary depending on, for example, the status of the network).

- Recommended file size (per scene) for 3D model files (obj/mtl)
  - obj file: 5 MB
  - mtl file: No recommended file size
- Texture file to be used for the 3D model [pixels]
  - File size: Unrestricted
  - Image texture size: The vertical and horizontal sizes must be an exponential of 2 (32, 64, 128, 256, 512, or 1024)

When determining the size of an obj file, consider the built-in memory size of the device to be used.

3.2 Notes on creating 3D models

Note the following when creating a 3D model:

- Set the size of the 3D model in metric units
  Ensure that the unit is [m] before exporting the 3D model to the obj and mtl files. If the unit is not [m], the 3D model may become overly big or small and may not be able to be overlaid properly.
- Output normal vector information of the 3D model
  After creating a 3D model, configure the settings so that normal vector information is output when the 3D model is exported to the obj and mtl files. If the obj file does not contain normal vector information, the 3D model will not be properly displayed.
- Do not change the name of the texture file set for the 3D model
  When you have export the 3D model to the obj and mtl files, the texture file name in the mtl file with have the name you set when creating the 3D model. Therefore, if you change the texture file name before registering the 3D model in the AR processing server, the texture file path defined in the mtl file becomes invalid and the texture will not be set properly for the 3D model.

3.2.1 Path format of mtl files

When you use a 3D editing tool to create a 3D model file (obj or mtl file) in a Windows operating system environment (on a Windows computer), the path of the texture defined in the "map_Ka/map_Kd" key of the mtl file may be in 8.3 format (a short file path format peculiar to Windows operating systems).
But Android and iOS (the operating environments of Interstage AR Processing Server) cannot analyze 8.3 format paths, and as a result the texture will not be rendered properly in the 3D model. Therefore, if the texture path defined in the mtl file uses the 8.3 format, change it to the image file name to be used as the texture of the 3D model.

If a texture path is changed to 8.3 format, the texture will not be rendered properly when the 3D model is displayed as augmented reality because Android and iOS (the operating environments of IARPS) cannot analyze 8.3 format paths.

Therefore, if the texture path defined in the mtl file uses the 8.3 format, change it to the image file name to be used as the texture of the 3D model.

3.2.2 Expressing materials and textures

If both material information and image texture are specified for a 3D model in Interstage AR Processing Server, the effect on the 3D model will be as follows:

- If a material shadow (Ka) is set
  There is no effect when the parameter is 0. Increasing (maximum: 1) the parameter highlights the body color of the 3D model.
- If the body color (Kd) of the material is set
  There is no effect when the parameter is 1. Decreasing (minimum: 0) the parameter makes the body color of the 3D model darker.
- If transparency (d/Tr) is set
  There is no effect when the parameter is 1. Decreasing (minimum: 0) the parameter increases the transparency of the body color of the 3D model.

Figure 3.1 When there is no change to material information

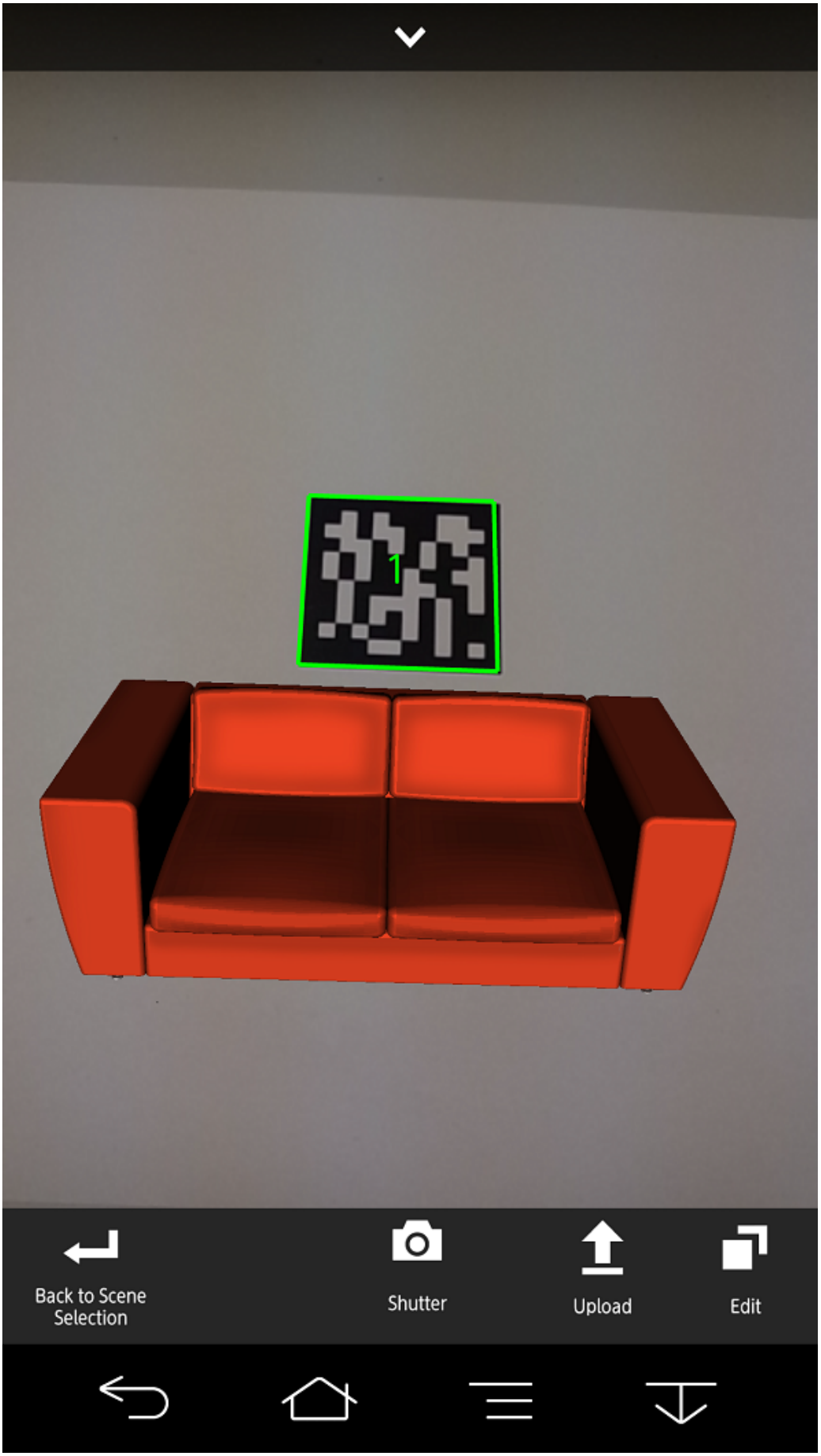


Figure 3.2 When Ka=1

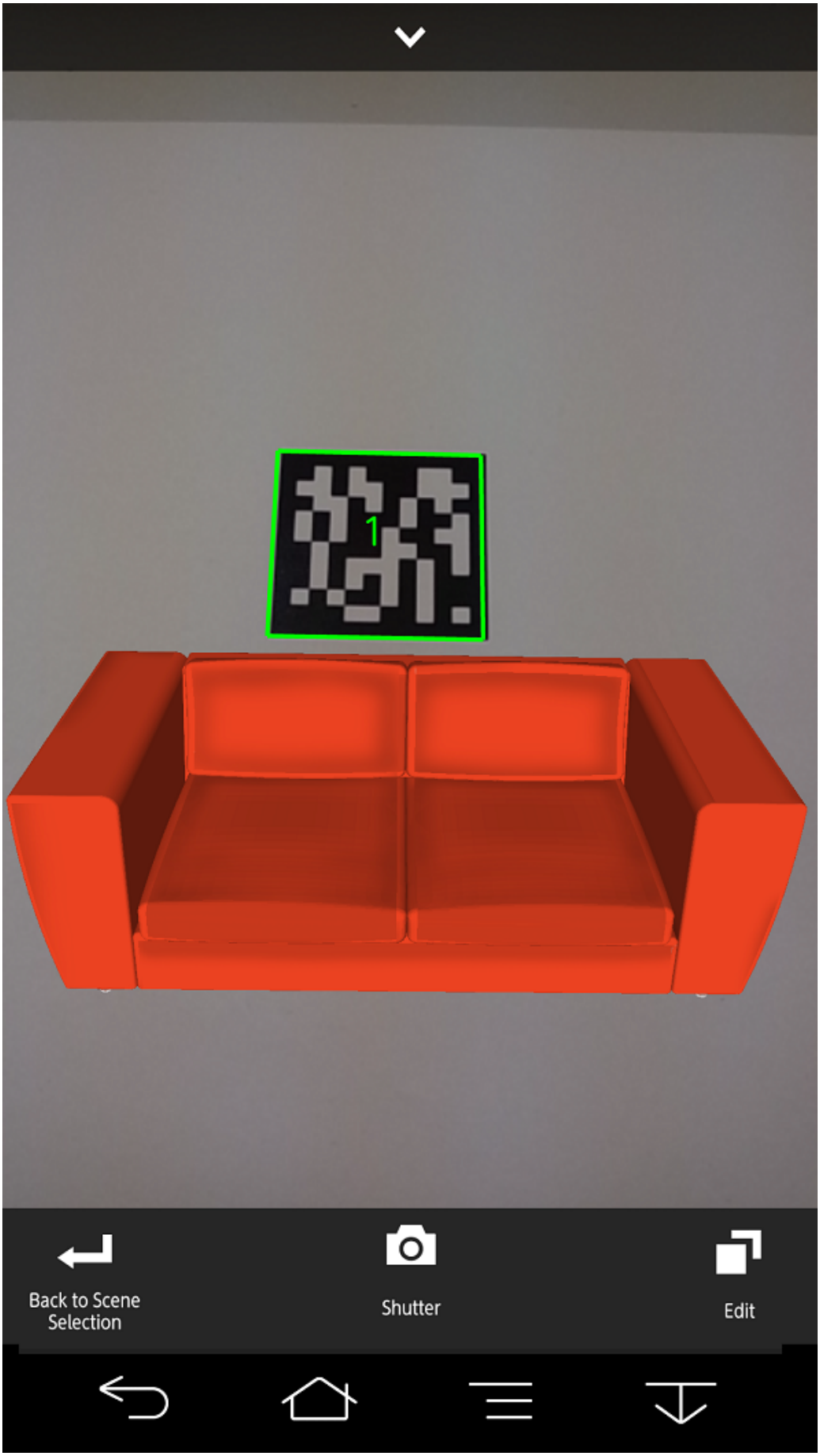


Figure 3.3 When Kd=0

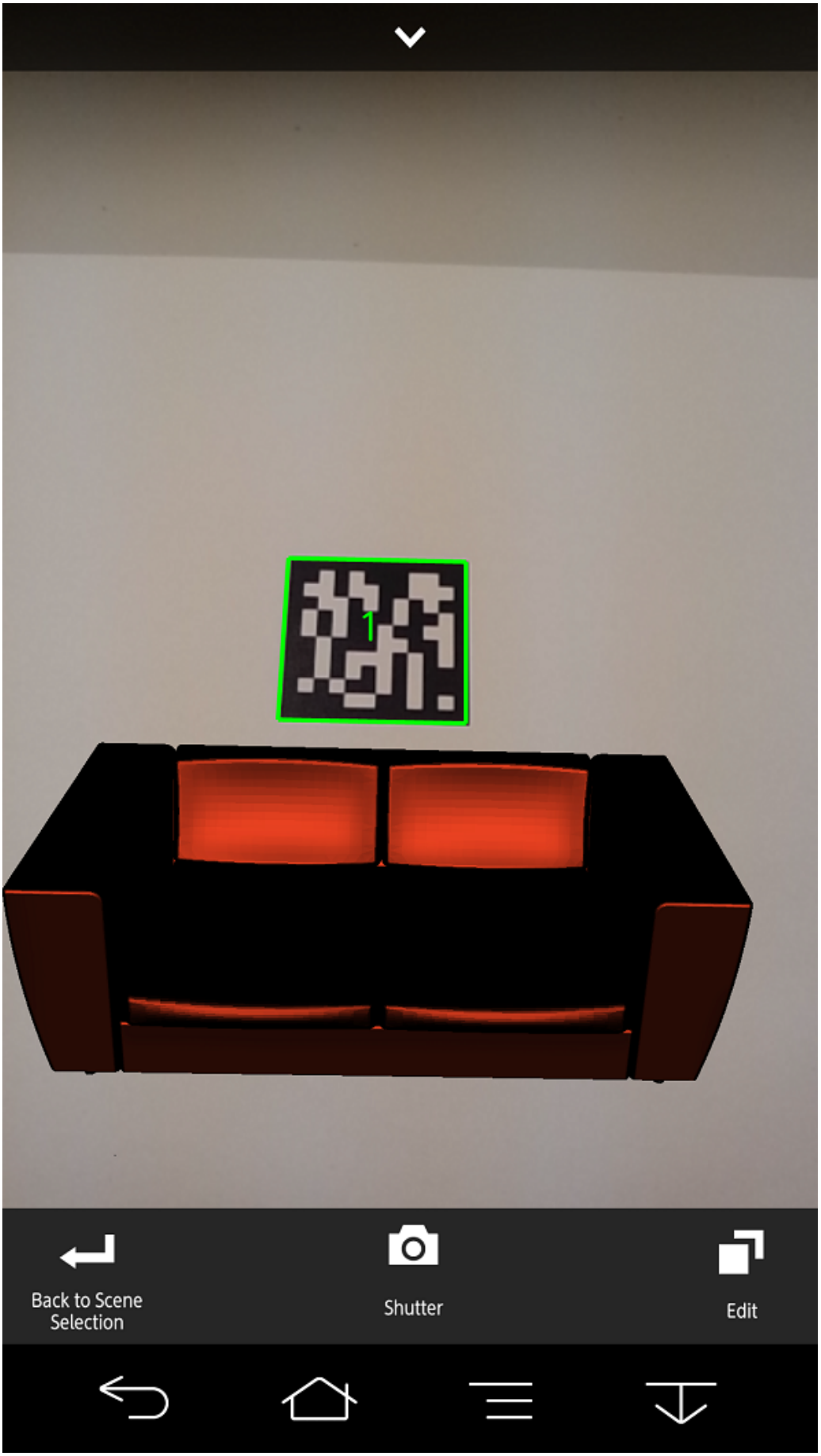
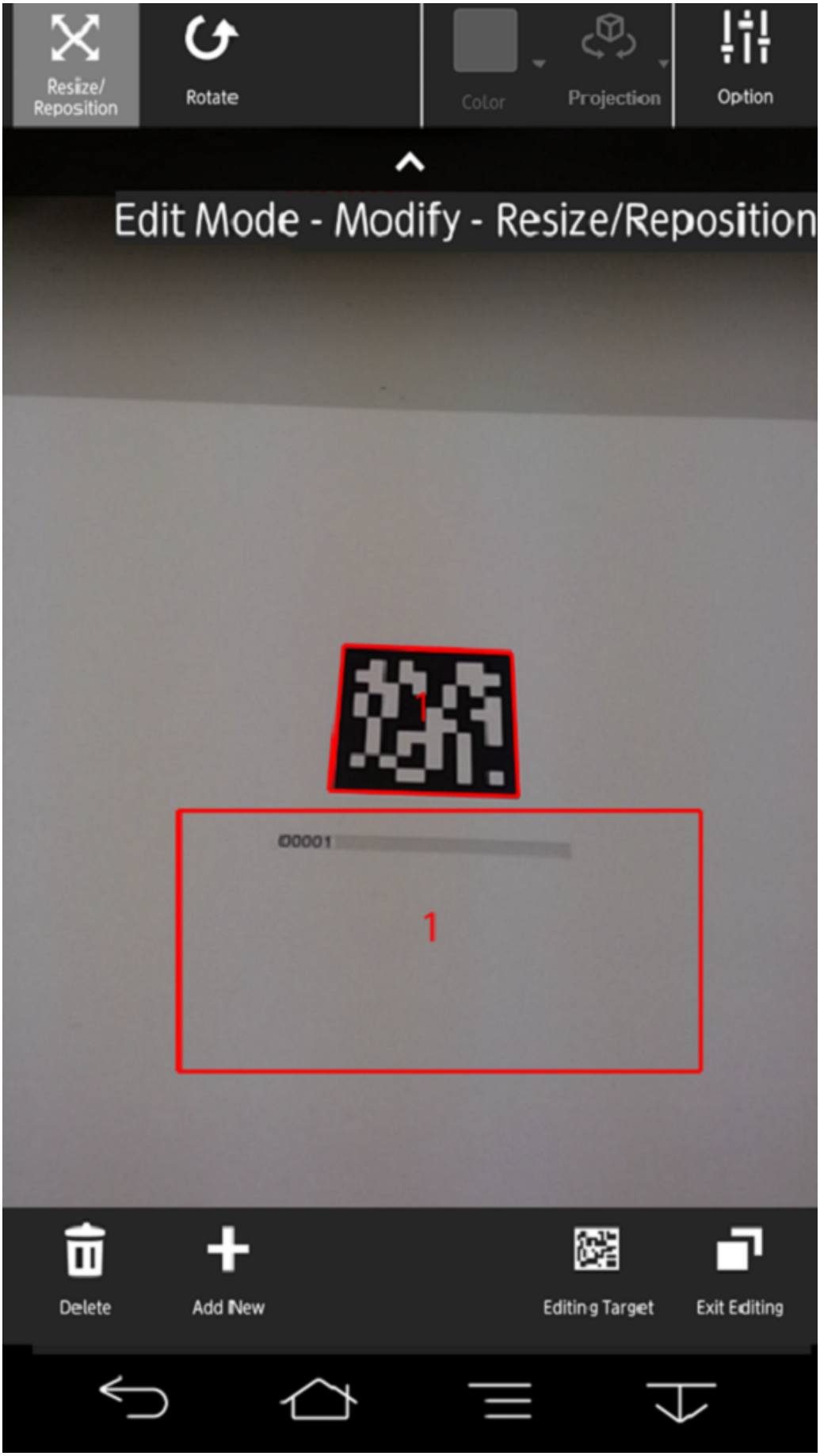


Figure 3.4 When d/Tr=0



3.2.3 Illumination settings

Illumination settings specified in a 3D editing tool are not output to the obj or mtl files. To apply shadow to a 3D model, use a texture image.

3.2.4 Size adjustment

If you are the AR marker recognition mode to display a 3D model, check the AR marker size and recognition distance, as well as the 3D model overlay magnification/reduction rate, and then adjust the size of the 3D model.

Marker size [mm]	50	100	500	1000
3D model size magnification/reduction rate [scaling factor]	1	2	10	20
Recognition distance [m]	1.4	2.8	14	28

Example

Assume that you are using the AR marker recognition mode to overlay a cube with sides of 10 m. If you use a marker with sides of 50 mm, the maximum marker recognition distance will be 1.4 m, so it may be difficult to display the 3D model within the screen.
In this case, using a marker with sides of 500 mm and a marker recognition distance of 14 m will enable you to manipulate the 3D model appropriately.
Before increasing the marker size, check the above-mentioned 3D model size magnification/reduction rate and adjust the 3D model size appropriately. In the example given here, reducing the 3D model size to a metric unit of 1 m per side will overlay a cube with sides of 10 m onto a real-world object when an AR marker with sides of 500 mm is recognized.

Chapter 4 Developing the overlay application (web application)

This chapter explains how to develop the web application, which is a component of the overlay application.

4.1 Overview

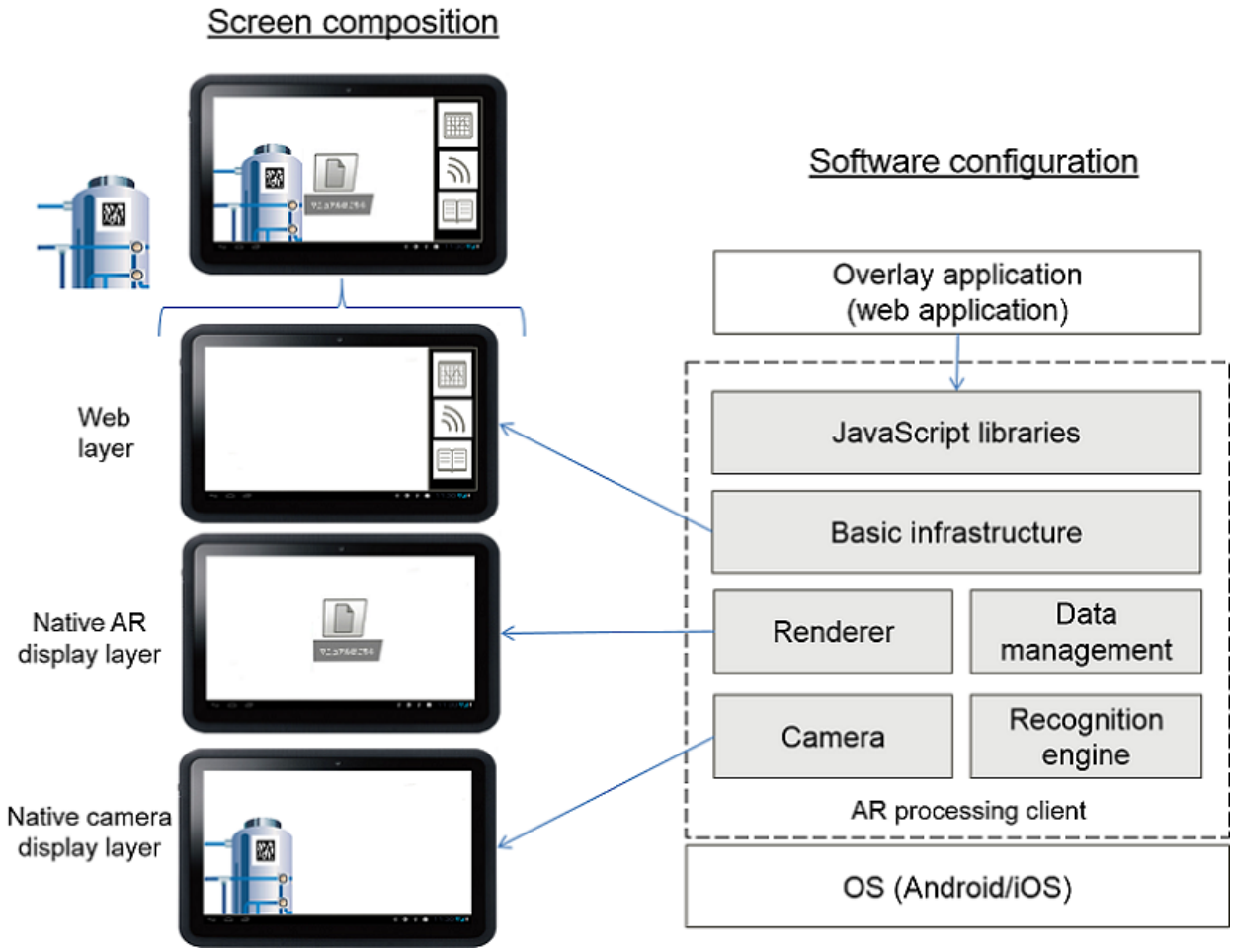
The web application is a component of the overlay application, which is developed for using AR technology to overlay the desired content on a camera preview screen. The web application must be developed using a web development language such as HTML or JavaScript. This application is downloaded from the server to a smart device and runs on the native application of the smart device.

4.1.1 Composition of the overlay application (web application)

Interstage AR Processing Server overlays digital information retrieved from the server onto real-world camera images. The screen that the overlay application provides to smart devices is rendered in the following three layers, with each layer being superimposed to form one screen:

1. Native camera display layer
   Displays the image captured by the camera.
2. Native AR display layer
   Displays the AR overlay content specified in the AR overlay definition.
3. Web layer
   Runs and displays the web application.

Figure 4.1 Example screen composition



The overlay application controls the content displayed in the web layer and in the native AR display layer. Development of the web layer is identical to development of general web applications and is therefore not explained in this manual. The following explains the development of the native AR display layer.

4.1.2 Operation mode

The overlay application has two operation modes: server connection mode and standalone mode.

Server connection mode

In this mode the overlay application communicates with the server to retrieve data. If the network status is offline, communication with the server fails. You can use saved data even in server connection mode.

Standalone mode

In this mode the overlay application uses data stored in offline storage instead of communicating externally. Attempts to upload data in this mode will fail - data can only be uploaded in server connection mode.

Operation mode	Network status	
	Online	Offline
Server connection mode	Download: Communicates with the server *Can store data Upload: Communicates with the server	Download: Fails *Can store data Upload: Fails
Standalone mode	Download: Uses storage Upload: Fails	Download: Uses storage Upload: Fails

Note

You can determine the operation mode only during the application startup. If you want to change it, you must close the application and then restart it.

4.1.3 Operating the camera

The camera starts when the overlay application starts. AR.Camera.startCameraView and AR.Camera.stopCameraView in the JavaScript library enable you to switch between camera start/stop. Change this setting to suit operation of the application.

If you cancel the application by pressing **Back** or **Home** on the device while the camera is stopped, the camera will start when you restart the application. You cannot use the overlay application to capture images using the camera.

Note

In the Windows version, stopCameraView() stops the camera and fasten the camera image currently.

4.1.4 Development flow

Follow the procedure below to develop the overlay application:

1. Develop the application to perform the following operations, in accordance with the requirements definition created during the design process:
   - Retrieve data from the AR processing server and registers it using the JavaScript library made publically available by Interstage AR Processing Server and the REST API.
   - Save resources when the site is offline.
   - Detect AR markers from camera images.
   - Pass an AR overlay definition based on the information in the detected AR marker and render the AR overlay content.
2. Create a war file of the developed overlay application.

4.1.5 Basic authentication

You can use basic authentication with overlay applications. When the user attempts to display a webpage for which basic authentication is enabled, a dialog box prompting the user to enter an account is displayed, and a valid account must be entered. To use Basic authentication automatically, click **Account Setting** and then **Add New Account** in the window for setting the overlay application, and set the account to be used. Set the account in **Host name:Port number**. You can set only one account in one **Host name:Port number**.

4.1.6 SSL communication

You can use SSL communication with overlay applications. If you access a server for which no SSL server certificate is registered, a warning dialog box indicating that the certificate is invalid is displayed. To prevent this warning, register in the server the SSL server certificate issued by the certificate authority. The method for registering a root certificate registered by default in a smart device or a new root certificate if a self-certificate was issued depends on the operating system and model. Therefore, check details before using these certificates.

4.1.7 Development environment

Prepare a development environment (including a text editor and Eclipse) required for development in a web development language such as HTML or JavaScript. Depending on the operating system, version, and model of the device, JavaScript standard APIs other than the JavaScript library provided by this product may not run, or may operate differently. Before starting development, check the APIs that can be run on your environment.

4.2 Overview of the JavaScript library

4.2.1 Features provided by the JavaScript library

The JavaScript library provided by Interstage AR Processing Server has the following features (refer to "JavaScript library reference" in the *Reference Guide* for details):

Library	Feature
Camera control (AR.Camera)	Start/stop camera
	Register/remove marker detection
	Retrieve ID and rectangular coordinates of detected marker
	Set/retrieve/check support of camera zoom value
	Take/retrieve photo
Log (AR.Log)	Output log
OS control (AR.OS)	Report tap coordinate to native
	Display specified URL content
	Perform online evaluation
	Retrieve/set recognition mode
	Exit AR overlay application
	Call settings screen
AR rendering (AR.Renderer)	Operate AR content: Retrieve
	Operate AR content: Add
	Operate AR content: Remove
	Operate AR content: Remove all
Location Data control (AR.Geolocation)	Register/remove Location Data detection
	Retrieve detected Location Data

Library	Feature
	Set/retrieve object display range
	Set radar for Location Data
	Retrieve camera tilt information/retrieval status
Data management control (AR.Data)	Save URL content
	Set AR processing server information
	Set use of offline storage
	Communicate with AR processing server: Retrieve
	Communicate with AR processing server: Register
	Communicate with AR processing server: Remove
	Retrieve URL of AR processing server
	Clear offline storage
Barcode control (AR.Barcode)	Register/remove barcode detection
	Retrieve ID and rectangular coordinate of barcode
Beacon control (AR.Bluetooth)	Register/remove region detection
	Retrieve the list of region UUIDs and beacons

4.2.2 Loading the JavaScript library

Load the ar.js file using the script tag of the HTML file where you want to use the JavaScript library.

Example

```
<script src="ar/ar.js"></script>
```

4.2.3 Calling APIs

Use JavaScript to call methods starting with "AR.". Refer to "JavaScript library reference" in the *Reference Guide* for details on APIs.

4.2.4 Retrieving API results

The JavaScript library APIs for calling the features of a native application are run asynchronously. APIs that perform asynchronous processing do not return values as methods, so return the results using the callback function. For APIs executed asynchronously, specify onSuccess (called when processing is successful) and onError (called when processing fails) as arguments.

A result object (Result class object) is returned by the callback function. The properties below can be used to retrieve the execution result and return value of the native application.

Property	Description
status	Status of the execution result of the native application. If execution is successful, "OK" is set, and if it fails, the error type is set. This property can be retrieved using getStatus().
value	The return value of the native application is set. The value to be set depends on the executed method and status. This property can be retrieved using getValue().

Example

Example of retrieving the JavaScript result

```
AR.Camera.startCameraView(onSuccess, onError);
onSuccess = function(result){
```

```
  alert(“Successful.") ;
};
onError = function(result){
  alert("Failed.\ n"+result.getStatus()+ " : " + result.getValue());
};
```

4.2.5 Notes

- HTML specifications dictate that data in the JavaScript layer is discarded when the screen is changed. Do not change the screen before the result callback is called.
- Manage application data not saved in the AR processing client on the web application side using web storage.
- The JavaScript prompt() function is used by the product, and therefore cannot be used by applications.
- During an unload or beforeunload event, smart devices with the following operating systems cannot execute APIs provided by the JavaScript library:
  - Android 4.4 or later
  - Windows
- The settings for the web layer features available from the web application are as follows:
  - Enabled
    - JavaScript
    - WebStorage
    - Application Cache
  - Disabled
    - Web SQL Database
    - Plug-ins (such as Flash)
    - Location Data
    - Saving form data
    - Saving passwords
    - File access (access to file:///)
- Consistency in the order of JSON object members between different operating systems is not guaranteed. For this reason, to access a member value, specify its key.

4.2.6 Error

The following error objects are thrown if an error occurs in the JavaScript library:

Property	Description
name	Name of the error
message	Error message
componentName	Name of the component in the JavaScript library where the error occurred
code	Error code
cause	Originating error

 See

Refer to the *Message Guide* for details on the error codes.

4.3 Data operation by the AR processing server

The REST API is used for connecting from the JavaScript library to the AR processing server. You can use the following APIs published by the AR processing server.

Item	Register (POST)	Identity via primary key API (GET)	Search (GET)	Delete (DELETE)
QType	N	Y	Y	N
QAttribute	N	Y	Y	N
QEntity	Y	Y	Y	Y
QValue	Y	Y	Y	Y
Quad	Y	N	Y	N

4.3.1 Setting information about the AR processing server

You must set information about the AR processing server in the AR processing client before you can access the AR processing server. There are two methods of doing this: setting information from the **Settings** window of the overlay application (native application), and using the JavaScript library. The following explains how to register information from the JavaScript library.

Registering from the JavaScript library

Register the information using AR.Data.setArServerInfo (_url, _id, _password, _onSuccess, _onError). The arguments are as follows:

Argument	Description
_url	URL of the AR processing server, excluding arsvdm (the final slash mark "/" is optional).
_id	ID if using Basic authentication on the AR processing server. Specify null if basic authentication is not being used.
_password	Password if using basic authentication on the AR processing server. Specify null if basic authentication is not being used.

Example

```
var url = "http://***.***.***.***:9102/";
var id = "aradmin";
var password = "aradmin"
AR.Data.setArServerInfo(url, id, password,  onSuccess, onError);
```

4.3.2 Registering data

You can register data using AR.Data.postArServerData (_query, _body, _onSuccess, _onError) of the JavaScript library. The data that can be registered is QEntity, QValue, and Quad.

Argument	Description
_query	URL path query to the AR processing server
_body	Data to be registered in the AR processing server. Specify a JSON string.

You can retrieve the response using the getValue() of the result object in the arguments of onSuccess.

Example

```
onSuccess = function(_result){
  var response = _result.getValue();
};
```

4.3.3 Examples of registering data

This section explains how to register each data instance, using the sample user-defined table usr_sample as an example.

4.3.3.1 Registering QEntity

The following explains how to register QEntity in usr_sample:

1. Create a JSON string for the QEntity to be registered.

```
var qentity = new Object();
qentity.qtypeName = "usr_sample";
var body = JSON.stringify(qentity);  // {"qtypeName" : "usr_sample"}
```

2. Register using AR.Data.postServerData().

```
AR.Data.postArServerData("qentities", body, onSuccess, onError);
```

3. The response when registration is successful is the JSON object of QEntity. An automatically assigned ID and version are appended - use them when registering or deleting QValue.

```
{"id":1, "qtypeName":"usr_sample", "version":1}
```

4.3.3.2 Registering QValue

Register a value in usr_name of usr_sample.

1. Create a JSON string for the QValue to be registered

```
var qvalue = new Object();
qvalue.qtypeName = "usr_sample";
qvalue.qentityId = 1; //Specify the ID of the registered qentity.
qvalue.qattributeName = "usr_name";
qvalue.stringValue = "user A";
qvalue.longValue = null;  //Specify null if the value is not to be used.
qvalue.floatValue = null;  //Specify null if the value is not to be used.
var body = JSON.stringify(qvalue);  // {"qtypeName":"usr_sample", "qentityId":1,
"qattributeName":"usr_name", "stringValue":"user A", "longValue":null, "floatValue":null}
```

2. Register using AR.Data.postServerData()

```
AR.Data.postArServerData("qvalues", body, onSuccess, onError);
```

3. The response when registration is successful is the JSON object of QValue. An automatically assigned version is appended - use it during deletion.

```
{"qtypeName":"usr_sample", "qentityId":1, "qattributeName":"usr_name", "stringValue":"user A",
"longValue":null, "floatValue":null, "version":1}
```

4.3.3.3 Registering Quad

The following explains how to register Quad in usr_sample:

1. Create QValue for each attribute. Do not specify qentityId and version, because they are assigned automatically.

```
var arId = new Object();
arId.qtypeName = "usr_sample";
arId.qattributeName = "ar_id";
arId.stringValue = null;
arId.longValue = 1;  //Specify null if the value is not to be used.
arId.floatValue = null;  //Specify null if the value is not to be used.
var arName = new Object();
arName.qtypeName = "usr_sample";
arName.qattributeName = "ar_name";
arName.stringValue = "inspection result";
```

```
arName.longValue = null;  //Specify null if the value is not to be used.
arId.floatValue = null;  //Specify null if the value is not to be used.

var arDescription = new Object();
arDescription.qtypeName = "usr_sample";
arDescription.qattributeName = "ar_description";
arDescription.stringValue = "Inspection result of the AR business sample application" ;
arDescription.longValue = null;
arDescription.floatValue = null;

var arRegistrationtime = new Object();
arRegistrationtime.qtypeName = "usr_sample";
arRegistrationtime.qattributeName = "ar_registrationtime";
arRegistrationtime.stringValue = null;
arRegistrationtime.longValue = new Date().getTime();
arRegistrationtime.floatValue = null;

var arModificationtime = new Object();
arModificationtime.qtypeName = "usr_sample";
arModificationtime.qattributeName = "ar_modificationtime";
arModificationtime.stringValue = null;
arModificationtime.longValue = new Date().getTime();
arModificationtime.floatValue = null;

var usrName = new Object();
usrName.qtypeName = "usr_sample";
usrName.qattributeName = "usr_name";
usrName.stringValue = "user A";
usrName.longValue = null;  //Specify null if the value is not to be used.
usrName.floatValue = null;  //Specify null if the value is not to be used.

var usrTemperature = new Object();
usrTemperature.qtypeName = "usr_sample";
usrTemperature.qattributeName = "usr_temperature";
usrTemperature.stringValue = null; //Specify null if the value is not to be used.
usrTemperature.longValue = 10;
usrTemperature.floatValue = null; //Specify null if the value is not to be used.
```

2. Create a JSON string for Quad. Do not specify ID and version, because they are assigned automatically.

```
var quad = new Object();
quad.qtypeName = "usr_sample";
quad.qvalues = [arId, arName, arDescription, arRegistrationtime, arModificationtime, usrName,
usrTemperature]; //Use the created attribute as an array.
var body  =  JSON.stringify(quad); //{"qtypeName" : "usr_sample"  ,  "qvalues" : [{"qtypeName" :
"usr_sample"  , "qattributeName"  :  "ar_id", ...}, ...]}
```

3. Register using AR.Data.postServerData()

```
AR.Data.postArServerData("quads", body, onSuccess, onError);
```

4. The response when registration is successful is the JSON object of Quad. An automatically assigned QEntityId and version are appended - use them during deletion.

4.3.4 Retrieving data

Retrieve data from the AR processing server using AR.Data.getArServerData(_query, _isSuperReload, _onSuccess, _onError) of the JavaScript library. The data that can be registered is QEntity, QValue, and Quad. The arguments are as follows:

Argument	Description
_query	URL query to the AR processing server
_isSuperReload	Whether to forcibly load from the AR processing server.

You can select either of the following methods for retrieving data:

- Specify the primary key of each data instance to retrieve specific data
- Specify conditions to retrieve multiple matching data instances

4.3.5 Examples of retrieving data

This section explains how to retrieve each instance, using the sample user-defined table usr_sample as an example.

4.3.5.1 Retrieving specific data using a primary key

Retrieving QType

1. Retrieve QType whose name is usr_sample.

```
var query = "qtypes/usr_sample";
AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the registered QType.

```
{"name":"usr_sample", "description":"business sample user data management table" , "version":
1}
```

Retrieving QAttribute

1. Retrieve QAttribute whose QType is usr_sample and name is usr_temperature.

```
var query = "qattributes/usr_sample/usr_temperature";
AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the registered QAttribute.

```
{"qtypeName":"usr_sample", "name":"usr_temperature", "description":"inspection temperature",
"orderIndex":7, "qvalueType":"LONG", "version":1}
```

Retrieving QEntity

1. Retrieve QEntity whose QType is usr_sample and ID is 1.

```
var query = "qentities/usr_sample/1";
AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the registered QEntity.

```
{"qtypeName":"usr_sample", "id":1, "version":1}
```

Retrieving QValue

1. Retrieve QValue whose QType is usr_sample, QEntity ID is 1, and attribute is usr_name.

```
var query = "qvalues/usr_sample/1/usr_name";
AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the registered QValue.

```
{"qtypeName":"usr_sample", "qentityId":1, "qattributeName":"usr_name", "stringValue":"user
A", "longValue":null, "floatValue":null, "version":1}
```

4.3.5.2 Retrieving data using a condition search

Searching for QType

The conditions specifiable for QType are as follows:

Query parameter	Description
type	Retrieved as a processing result. - RECORDS=Returns the result array - COUNT=Returns the total number of results - RECORDSANDCOUNT=Retrieves the result array and total number of results
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}
nameRanges	name range array Example: [{"start":"AR100","end":"AR999"}]
sortOrders	Sort order array - can have the following values: - NAME = Returns name in ascending order - NAME_DESC = Returns name in descending order Example: ["NAME"]

1. Searching for QType is as follows:

```
var query = "qtypes?";
//type specification
query += "type=RECORDSANDCOUNT";  //Retrieve the result array and total number of results

//limitRange specifications
var limitRange = new Object();
limitRange.start = 1;
limitRange.end = 10;
query += "&limitRange="+JSON.stringify(limitRange); // {"start":1, "end":10}

//nameRanges specifications
var nameRanges = new Array();
nameRanges[0] = new Object()
nameRanges[0].start = "usr_sample";
nameRanges[0].end = "usr_sample100"
query += "&nameRanges=" +JSON.stringify(nameRanges); //{"start":"usr_sample",
"end":"usr_sample100"}

//sortOrders specifications
var sortOrders = ["NAME_DESC"];
query += "&sortOrders" + JSON.stringify(sortOrders);
AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the QType that matches the condition.

```
{unlimitedRecordCount:1, "records":[searchResultQtypeArray]}
```

Searching for QAttribute

The conditions searchable for QAttribute are as follows:

Query parameter	Description
type	Retrieved as a processing result. - RECORDS=Returns the result array - COUNT=Returns the total number of results

Query parameter	Description
	- RECORDSANDCOUNT=Retrieves the result array and total number of results
limitRange	Range of records in the search result to return. Must be greater than 0. Example: {"start":1,"end":10}
qtypeNameRanges	name range array Example: [{"start":"AR100","end":"AR999"}]
nameRanges	name range array Example: [{"start":"remarks01","end":"remarks99"}]
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"]

1. Searching for QAttribute is as follows:

```
var query = "qattributes?";
//type specification
query += "type=RECORDSANDCOUNT";  //Retrieve the result array and total number of results

//limitRange specifications
var limitRange = new Object();
limitRange.start = 1;
limitRange.end = 10;
query += "&limitRange="+JSON.stringify(limitRange); // {"start":1, "end":10}

//qtypeNameRanges specifications
var qtypeNameRanges = new Array();
qtypeNameRanges[0] = new Object;
qtypeNameRanges[0].start = "usr_sample";
qtypeNameRanges[0].end = "usr_sample100"
query += "&qtypeNameRanges=" +JSON.stringify(qtypeNameRanges); //{"start":"usr_sample",
"end":"usr_sample100"}

//nameRanges specifications
var nameRanges = new Array();
nameRanges[0] = new Object();
nameRanges[0].start = "usr_";
nameRanges[0].end = "usr_z"
query += "&nameRanges=" +JSON.stringify(nameRanges); //{"start":"usr_", "end":"usr_z"}

//sortOrders specifications
var sortOrders = ["QTYPENAME_DESC","ORDERINDEX"];
query += "&sortOrders=" + JSON.stringify(sortOrders); //["QTYPENAME_DESC", "ORDERINDEX"]

AR.Data.getArServerData(query, true, onSuccess, onError);

nameRanges[0] = new Object();
```

```
nameRanges[0].start = "usr_";
nameRanges[0].end = "usr_z"
query += "&nameRanges=" +JSON.stringify(nameRanges); //{"start":"usr_", "end":"usr_z"}

//sortOrders specifications
var sortOrders = ["QTYPENAME_DESC","ORDERINDEX"];
query += "&sortOrders=" + JSON.stringify(sortOrders); //["QTYPENAME_DESC", "ORDERINDEX"]

AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the QType that matches the condition.

```
{unlimitedRecordCount:1, "records":[searchResultQattributeArray]}
```

Searching for QEntity

The conditions searchable for QEntity are as follows:

Query parameter	Description
type	Retrieved as a processing result. - RECORDS=Returns the result array - COUNT=Returns the total number of results - RECORDSANDCOUNT=Retrieves the result array and total number of results
limitRange	Range of records in the search result to return. Must be greater than 0. Example: {"start":1,"end":10}
qtypeNameRanges	name range array Example: [{"start":"AR100","end":"AR999"}]
idRanges	id range array Example: [{"start":100,"end":200}]
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"]

1. Searching for QEntity is as follows:

```
var query = "qentities?";
//type specification
query += "type=RECORDSANDCOUNT";  //Retrieve the result array and total number of results
//limitRange specifications
var limitRange = new Object();
limitRange.start = 1;
limitRange.end = 10;
query += "&limitRange="+JSON.stringify(limitRange); // {"start":1, "end":10}

//qtypeNameRanges specifications
var qtypeNameRanges = new Array();
qtypeNameRanges[0] = new Object();
qtypeNameRanges[0].start = "usr_sample";
qtypeNameRanges[0].end = "usr_sample100"
```

```
query += "&qtypeNameRanges=" +JSON.stringify(qtypeNameRanges); //{"start":"usr_sample",
"end":"usr_sample100"}

//idRanges specifications
var idRanges = new Array();
idRanges[0] = new Object();
idRanges[0].start = 1;
idRanges[0].end = 10000000000000000;
query += "&idRanges=" +JSON.stringify(idRanges); //{"start":1, "end":10000000000000000}

//sortOrders specifications
var sortOrders = ["QTYPENAME_DESC","ID"];
query += "&sortOrders" + JSON.stringify(sortOrders); //["QTYPENAME_DESC", "ORDERINDEX"]

AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the QType that matches the condition.

```
{unlimitedRecordCount:1, "records":[searchResultQentityArray]}
```

Searching for QValue

The conditions specifiable in a QValue search are as follows:

Query parameter	Description
type	Retrieved as a processing result. - RECORDS=Returns the result array - COUNT=Returns the total number of results - RECORDSANDCOUNT=Retrieves the result array and total number of results
limitRange	Range of records in the search result to return. Must be greater than 0. Example: {"start":1,"end":10}
qtypeNameRanges	name range array Example: [{"start":"AR100","end":"AR999"}]
qentityIdRanges	id range array Example: [{"start":100,"end":200}]
qattributeNameRanges	name range array Example: [{"start":"AR100","end":"AR999"}]
stringValueRanges	stringValue range array Example: [{"start":"Kanagawa Prefecture_A","end":"Kanagawa Prefecture_Z"}]
longValueRanges	longValue range array Example: [{"start":1,"end":100}]
floatValueRanges	floatValue range array Example: [{"start":0.1,"end":0.9}]
sortOrders	Sort order array - can have the following values: - QTYPENAME = Returns qtypeName in ascending order - QTYPENAME_DESC = Returns qtypeName in descending order - QENTITYID = Returns qentityId in ascending order

Query parameter	Description
	- QENTITYID_DESC = Returns qentityId in descending order - QATTRIBUTENAME = Returns qattributeName in ascending order - QATTRIBUTENAME_DESC = Returns qattributeName in descending order Example: ["QTYPENAME","QENTITYID","QATTRIBUTENAME"]

1. Searching for QValue is as follows:

```
var query = "qvalues?";
//type specification
query += "type=RECORDSANDCOUNT";  //Retrieve the result array and total number of results

//limitRange specifications
var limitRange = new Object();
limitRange.start = 1;
limitRange.end = 10;
query += "&limitRange="+JSON.stringify(limitRange); // {"start":1, "end":10}

//qtypeNameRanges specifications
var qtypeNameRanges = new Array();
qtypeNameRanges[0] = new Object();
qtypeNameRanges[0].start = "usr_sample";
qtypeNameRanges[0].end = "usr_sample100"
query += "&qtypeNameRanges=" +JSON.stringify(qtypeNameRanges); //{"start":"usr_sample",
"end":"usr_sample100"}

//qentityIdRanges specifications
var qentityIdRanges = new Array();
qentityIdRanges[0] = new Object();
qentityIdRanges[0].start = 1;
qentityIdRanges[0].end = 10000000000000000;
query += "&qentityIdRanges=" +JSON.stringify(qentityIdRanges); //{"start":1, "end":
10000000000000000}

//qattributeNameRanges specifications
var qattributeNameRanges = new Array();
qattributeNameRanges[0] = new Object();
qattributeNameRanges[0].start = "usr_";
qattributeNameRanges[0].end = "usr_z"
query += "&qattributeNameRanges=" +JSON.stringify(qattributeNameRanges); //{"start":"usr_",
"end":"usr_z"}

//longValue specifications
var longValueRanges = new Array();
longValueRanges[0] = new Object();
longValueRanges[0].start = 0;
longValueRanges[0].end = 20;
query += "&longValueRanges=" +JSON.stringify(longValueRanges); //{"start":1, "end":20}
//sortOrders specifications
var sortOrders = ["QTYPENAME_DESC","QATTRIBUTENAME"];
query += "&sortOrders" + JSON.stringify(sortOrders); //["QTYPENAME_DESC", "QATTRIBUTENAME"]

AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the QValue that matches the condition.

```
{unlimitedRecordCount:1, "records":[searchResultQvalueArray]}
```

Searching for Quad

The conditions searchable for Quad are as follows:

Query parameter	Description
type	Retrieved as a processing result. - RECORDS=Returns the result array - COUNT=Returns the total number of results - RECORDSANDCOUNT=Retrieves the result array and total number of results
limitRange	Range of records in the search result to return. Must be greater than 0. Example: {"start":1,"end":10}
qattributeOrderIndexRange	Range for retrieving the result array in the QAttribute direction. Specify in the sequence used for sorting QAttribute.orderIndex in ascending order. Must be greater than 0. Example: {"start":1,"end":10}
qtypeNameRanges	name range array Example: [{"start":"AR100","end":"AR999"}]
whereExpressions	Filter condition array
sortOrderExpressions	Sort condition array

Expressions thrown by Quad are as follows:

Attribute	Description
qattributeNameRanges	Same as QValue Example: [{"start":"column01","end":"column09"}] If a range such as [{"start":"column01","end":"column99"}] is specified as a 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 specify a range or range array.
qvalueType	QAttribute.qvalueType identified by QValue.qtypeName, .qentityId, and .qattributeName
qvalueRanges	QValue.xxxValue range
desc	Whether to sort in descending order. Effective only when there is a sort condition.

1. Searching for Quad is as follows:

```
var query = "quads?";
//type specification
query += "type=RECORDSANDCOUNT";  //Retrieve the result array and total number of results
//limitRange specifications
var limitRange = new Object();
limitRange.start = 1;
limitRange.end = 10;
query += "&limitRange="+JSON.stringify(limitRange); // {"start":1, "end":10}

//qattributeOrderIndexRange specifications
var qattributeOrderIndexRange = new Object();
qattributeOrderIndexRange.start = 1;
qattributeOrderIndexRange.end = 10;
```

```
query += "&limitRange="+JSON.stringify(qattributeOrderIndexRange); // {"start":1, "end":10}

//qtypeNameRanges specifications
var qtypeNameRanges = new Array();
qtypeNameRanges[0] = new Object();
qtypeNameRanges[0].start = "usr_sample";
qtypeNameRanges[0].end = "usr_sample100"
query += "&qtypeNameRange=" +JSON.stringify(qtypeNameRanges); //{"start":"usr_sample",
"end":"usr_sample100"}

//whereExpressions specifications
var whereExpressions = new Array();
whereExpressions[0] = new Object();
whereExpressions[0].qattributeRanges = new Array();
whereExpressions[0].qattributeRanges[0] = new Object();
whereExpressions[0].qattributeRanges[0].start = "usr_";
whereExpressions[0].qattributeRanges[0].end = "usr_z";
whereExpressions[0].qvalueType = "LONG";
whereExpressions[0].qvalueRanges = new Array();
whereExpressions[0].qvalueRanges[0] = new Object();
whereExpressions[0].qvalueRanges[0].start = 0;
whereExpressions[0].qvalueRanges[0].end = 20;
query += "&whereExpressions="+JSON.stringify(whereExpressions);

//sortOrderExpressions specifications
var sortOrderExpressions = new Array();
sortOrderExpressions[0] = new Object();
sortOrderExpressions[0].qattributeRanges = new Array();
sortOrderExpressions[0].qattributeRanges[0] = new Object();
sortOrderExpressions[0].qattributeRanges[0].start = "usr_temperature";
sortOrderExpressions[0].qattributeRanges[0].end = "usr_temperature";
sortOrderExpressions[0].qvalueType = "LONG";
sortOrderExpressions[0].qvalueRanges = new Array();
sortOrderExpressions[0].qvalueRanges[0] = new Object();
sortOrderExpressions[0].qvalueRanges[0].start = 0;
sortOrderExpressions[0].qvalueRanges[0].end = 20;
sortOrderExpressions[0].desc = true;
query += "&sortOrderExpressions="+JSON.stringify(sortOrderExpressions);

AR.Data.getArServerData(query, true, onSuccess, onError);
```

2. The response is the JSON object of the Quad that matches the condition.

```
{unlimitedRecordCount:1, "records":[searchResultQuadArray]}
```

4.3.6 Deleting data

Delete data using AR.Data.deleteArServerData (_query, _version, _onSuccess, _onError) of the JavaScript library. The data that can be deleted is QEntity and QValue. The arguments are as follows:

Argument	Description
_query	URL path query to the AR processing server
_version	Version of the data, currently registered in the AR processing server, to be deleted.

Example

The following explains how to delete each instance, using the sample user-defined table usr_sample.

Deleting QEntity

To delete QEntity with ID of 1 and version of 1 registered in usr_sample:

```
AR.Data.deleteArServerData("qentities/usr_sample/1", 1, onSuccess, onError);
```

Deleting QValue

To delete QValue with qentity ID of 1, QAttribute of usr_name, and version of 1 registered in usr_sample:

```
AR.Data.deleteArServerData("qvalues/usr_sample/1/ usr_name", 1, onSuccess, onError);
```

4.4 Offline operation

The overlay application supports three types of resources: web resources, resources saved in the AR processing server, and external resources. To use these resources offline, you must download and save them while online, but different methods are used for saving each resource type.

Resource type	Save method
Web resources	HTML cache feature
Resources in the AR processing server	Offline storage or web storage
External resources for AR overlay content	Offline storage

4.4.1 Web resources

4.4.1.1 Saving web resources

You can save web resources such as HTML files, CSS files, and image files using the HTML 5 Application Cache mechanism, The cannot be saved using the AR application. When an HTML file for which appcache has been set is loaded, the specified files are automatically downloaded. Download resources before going offline.

Example

Example appcache file

```
CACHE MANIFEST
#ver 20130702_0001
#Specify the resources to be cached.
CACHE:
index.html
prework.html
work.html
comment.html
gyoumu.css
apl_sample/apl.js
ar/ar.js
#Resources required to connect to the server.
NETWORK:
#Alternative resources to use when unable to access the main resources.
FALLBACK:
```

Example setting in HTML

```
<html manifest="work.appcache">
```

Note

- Resource downloading using an appcache file may fail (due to communication problems, for example). For this reason, start the device while offline and confirm that the resources were downloaded successfully before using them.
- If using Android 4.4 or later, iOS8 or later, or Windows, when the following conditions are met:
  - HTTPS is being used, and
  - A self-certificate is being used as the SSL server certificate for the application server, and
  - The certificate to be used by the server has not been installed on the device.

  appcache may not run properly except if an official server certificate is used or a self-certificate is installed to the device.

4.4.1.2 Updating web resources

During startup, applications that use the Application Cache mechanism check the appcache file deployed to the server.
If the appcache file has been updated, then they update the device cache.
If the appcache file has not been updated but other files have, the updates are not reflected in the device cache.
If the device cache is not updated even though the appcache file has been updated, an error may have occurred in appcache on the device.
If this happens, delete the web resources and then retrieve the application cache again.
If HTML files, JavaScript files and other web resources are frequently updated during development of a web application, deleting the appcache settings eliminates the need to update the appcache file.

Note

If using the Android or Windows version and a web resource has been updated, terminate the application processes and tap **Delete web cache** from the overlay application configuration screen, and then retrieve the application cache again.

4.4.2 Resources in the AR processing server

4.4.2.1 Saving data from the AR processing server

There are two methods of handling AR processing server data in standalone mode: retrieving all data to be used and saving it in web storage in advance, or using offline storage. If the data volume is large, memory in the web layer may run out, so we recommend using offline storage.

- Web storage
  This method involves saving data to the local environment using the features of HTML 5. After retrieving data from the AR processing server, process it as required, and then save it. There are two types of storage: sessionStorage (which is available only during a session), and localStorage (which saves data permanently in the local environment). Save data using the format setItem(key,value), and retrieve it using getItem(key). Refer to HTML 5 specifications for details on how to use web storage.
- Offline storage
  This method involves saving data in the memory of the AR processing client. Enabling offline storage causes the response of AR.Data.getArServerData() executed in server connection mode to be saved in offline storage, with a query used as the key. You can retrieve data saved in offline storage by executing AR.Data.getArServerData using the same query.
  Use AR.Data.useOfflineStorage() to set offline storage.

```
AR.Data.useOfflineStorage(true, onSuccess, onError);
```

Point

When offline storage is enabled, data will be retrieved from offline storage even while the device is online, unless you forcibly load data from the server.

Executing a query that is not saved in offline storage will cause an error.

4.4.2.2 Updating locally saved data

- Web storage
  Execute setItem(key, value) for the key to be updated, in the same manner as when saving data.

4.4.2.3 Deleting locally saved data

- Web storage
  To delete data with the specified key, execute removeItem(key). To delete all data saved in web storage, execute clear().
- Offline storage
  Use AR.Data.clearResourceStorage() to delete data from offline storage.

```
AR.Data.clearResourceStorage(onSuccess, onError);
```

4.4.3 External resources for AR overlay content

If a resource used by an image texture or called by a tapAction is an external resource (a resource used via a network by specifying an URL), you cannot download it in standalone mode or in offline state. Only data registered in the AR processing server is saved in offline storage by AR.Data.getArServerData().

To save external resources in offline storage, use AR.Renderer.put() in server connection mode - this will register AR overlay definition data that uses external resources in the native AR display layer. When AR overlay definition data is analyzed, the AR native application downloads the external resources and saves them in offline storage. Alternatively, you can save resources by using AR.Data.cacheUrlResource().

4.5 Marker detection

Use AR.Camera.addMarkerListener() for confirming that the marker is within the camera image and for finding out which marker is being captured.

Point

A marker detection event is generated each time a marker is detected in the camera image or lost from it. Therefore, by registering the callback function beforehand, you can set processing to be performed when a marker is detected or lost.

Example

```
AR.Camera.addMarkerListener(onSuccess, onError, onSuccessListener, onErrorListener);
var listenerId;
//Callback function to be called during registration of a marker detection event listener
onSuccess = function(_result){
  listenerId = _result.getValue();
  alert("Successfully registered the marker detection event listener.") ;
}
onError = function(_result){
  alert("Failed to register the marker detection event listener.\ n" + _result.getStatus() +" : " +
  _result.getValue());
};
//Callback function to be called during generation of a marker detection event
onSuccessListener = function(_result){
  //If a marker is detected
  if(_result.getValue().status == true) {
    alert("Marker"+_result.getValue().value+"was detected."   );
  //If a marker is lost
  } else if(_result.getValue().status == false){
    alert("Marker"+_result.getValue().value+"was lost.") ;
  }
};
```

```
onErrorListener = function(_result){
  alert("Failed to detect a marker.\ n + _result.getStatus() +" : " + _result.getValue());
};
```

To delete a registered marker detection event listener, use AR.Camera.removeMarkerListener(), specifying the listener ID to be deleted.

Example

```
AR.Camera.removeMarkerListener(listenerId, onSuccess, onError);

onSuccess = function(_result){
  alert("Successfully deleted the marker detection event listener.") ;
}
onError = function(_result){
  alert("Failed to delete the marker detection event listener.\ n" + _result.getStatus() +" : " +
  _result.getValue());
};
```

Note

- Changing an HTML screen deletes all event listeners that were registered using AR.Camera.addMarkerListener(). Add a process that will, for example, re-register the event listeners when the screen is changed.
- When performing the process below relating to the marker coordinates, do not use a scroll bar or frame in the web page. Marker detection information is retrieved to suit the displayed web page.
  - Tap action
  - Retrieve the marker coordinates by using getCurrentMarkers()

4.6 Back button

To return to the previous HTML screen when displaying HTML screens in the overlay application, press the back button on an Android device or the <B>Back</B> button soft-key on an iOS or Windows device.

Pressing the back button on the first screen that loads from the AR native application closes the overlay application and returns to the AR native application.

4.7 Displaying AR overlay content

4.7.1 Creating AR overlay definitions

To display overlays on an AR overlay application, you must set an AR overlay definition in the native AR display layer using the JavaScript library. There are two methods of creating an AR overlay definition: creating one from scratch by using a JavaScript library, and converting data already registered in the AR processing server.

4.7.1.1 Creating definitions using a class of the JavaScript library

AR overlay definitions are expressed using the AR.Renderer.SuperimposedGraphic class of the JavaScript library. Create an AR.Renderer.SuperimposedGraphic object and set the required property.

Example

- Creating a text texture

```
//Create an AR overlay definition object.
var superimposedGraphic  = new AR.Renderer.SuperimposedGraphic();
//Set the projection type.
superimposedGraphic.setProjectionType(AR.Renderer.SuperimposedGraphic.ProjectionType.ORTHO2D);

//Create a translation coordinate object.
var translation = new AR.Renderer.Point();
translation.setX(1);
translation.setY(1);
translation.setZ(1);
//Set the translation coordinates for the AR overlay definition.
superimposedGraphic.setTranslation(translation);

//Create a rotation coordinate object.
var rotation = new AR.Renderer.Point();
rotation.setX(0); //x-coordinate direction
rotation.setY(0); //y-coordinate direction
rotation.setZ(0); //z-coordinate direction
//Set the rotation coordinates for the AR overlay definition.
superimposedGraphic.setRotation(rotation);

//Create a tap action (URL action) object.
var urlAction = new AR.Renderer.URLAction();
urlAction.setSrc("http://_"); //Specify the URL to be operated by the tap action.
//Set the tap action in the AR overlay definition.
superimposedGraphic.setTapAction(urlAction);

//Create a graphic object.
var squareModelGraphic = new AR.Renderer.SquareModelGraphic();
//Create a scale object.
var scale = new AR.Renderer.Point();
scale.setX(1); //x-coordinate direction
scale.setY(1); //y-coordinate direction
scale.setZ(1); //z-coordinate direction
//Set the scale for the graphics.
squareModelGraphic.setScale(scale);

//Create a texture object (text texture).
var textTexture = new AR.Renderer.TextTexture();
textTexture.setBackgroundColor(4294951104);
textTexture.setColor(4289724416);
textTexture.setFontSize(20);
textTexture.setText(“text”);
textTexture.setWordWrap(true);
//Create a size object.
var size = new AR.Renderer.Size();
size.setWidth(160); //Texture width
size.setHeight(80); //Texture height
//Set a size for the texture.
textTexture.setSize(size);

//Set a texture for the graphics.
squareModelGraphic.setTexture(textTexture);

//Set graphics for the AR overlay definition.
superimposedGraphic.setGraphic(squareModelGraphic);
```

- Creating an image texture

  If using an image, create an image texture object and set it for the graphics.

```
//Create a texture object (image texture).
var imageTexture = new AR.Renderer.ImageTexture();
imageTexture.setSrc("http://_"); //Specify the URL of the image to be overlaid.
```

```
//Set the texture for the graphics.
squareModelGraphic.setTexture(imageTexture);
```

4.7.1.2 Retrieving data from the AR processing server

When retrieving or searching for QValue, or when searching for Quad, retrieve the JSON string registered in the arpoi_superimposedgraphic attribute of the arpoiarmk_default table or arpoigps_default table. Use AR.Renderer.parseSuperimposedGraphic() to convert to an AR.Renderer.SuperimposedGraphic object.

See

Refer to "4.3 Data operation by the AR processing server" for details on how to retrieve data from AR data management.

4.7.1.3 Identifying AR overlay definitions

To specify an AR overlay definition and change the content of the overlay, or to display only the specified AR overlay definition, you must identify the target AR overlay definition. Each AR overlay definition ID has a unique value, and is assigned automatically when an AR overlay definition is created on the data management console. To use your own values, give a unique name to each AR overlay definition.

4.7.1.4 Embedding user-defined data

To change the display of AR overlay content using user-defined data, edit the elements to be changed of the AR overlay definition created as the AR.Renderer.SuperimposedGraphic object.

Example

- To change the text of AR overlay content:

```
//SuperimposedGraphic -> SquareModelGraphic -> Edit Text in TextTexture
SuperimposeGraphic.getGraphic().getTexture().setText("stringToBeDisplayed");
```

- To change the image of AR overlay content:

```
//SuperimposedGraphic -> SquareModelGraphic -> Edit Src in ImageTexture
SuperimposeGraphic.getGraphic().getTexture().setSrc("urlOfImageToBeDisplayed");
```

4.7.2 Setting AR overlay definitions

Set AR overlay definitions you have created with their coordinate system objects in the AR processing client. If using AR markers, generate an AR.Renderer.FJARMarkerCoordinateSystem object and set the AR marker ID. If using Location Data, generate an AR.Renderer.FJGeolocationCoordinateSystem object and set in accordance with the coordinate system.

Example

- If using AR markers

```
//Create a coordinate system object
var coordinateSystem = new AR.Renderer.FJARMarkerCoordinateSystem();
//Set the marker ID
coordinateSystem.setValue(1);

AR.Renderer.put(coordinateSystem, superimposedGraphic, onSuccess, onError);
```

- If using Location Data

```
//Create a coordinate system object
var coordinateSystem = new AR.Renderer.FJGeolocationCoordinateSystem();

AR.Renderer.put(coordinateSystem, superimposedGraphic, onSuccess, onError);
```

- If using barcodes

```
//Create a coordinate system object
var coordinateSystem = new AR.Renderer.FJBarcodeCoordinateSystem();
//Set the barcode ID
coordinateSystem.setValue("http://www.example.com");
AR.Renderer.put(coordinateSystem, superimposedGraphic, onSuccess, onError);
```

- If using beacons

```
//Create a coordinate system object
var coordinateSystem = new AR.Renderer.FJBeaconCoordinateSystem();
//Set a UUID (8-4-4-4-12)
coordinateSystem.setValue("123e4567-e89b-12d3-a456-426655440000");
AR.Renderer.put(coordinateSystem, superimposedGraphic, onSuccess, onError);
```

4.7.2.1 Switching the AR overlay content

The AR processing client simultaneously renders all AR overlay definitions set for the same marker. The AR processing client does not manage scenarios and scenes and does not distinguish between them. Therefore, to switch the display according to the scenario and scene, change the AR overlay definition when the display is switched.

4.7.3 Tap action

To run tapAction set as the tap action for AR overlay content, use AR.OS.onBodyClick() - this will propagate the tap event on the HTML layer to the native AR display layer. You cannot operate the native AR display layer directly because it is covered by the WebView layer. If AR content exists on the reported coordinates, the set tapAction is run.

 Example

Example of registering a tap event

```
//Register the screen tap event listener.
if(window.navigator.userAgent.match(/(iPad|iPhone|iPod)/i)) //On iOS:
  document.addEventListener("touchstart", bodyclick, false);
else
  document.addEventListener("click", bodyclick, false); //On android or Windows:

bodyclick = function(){
  //Report the tap coordinates to the native AR display layer.
  AR.OS.onBodyClick(event, onSuccess, onError);
};
```

4.7.4 Setting coordinate positions

- In the Android and iOS versions, the tap event coordinates may become misaligned if the device screen size and the HTML size do not match. Set viewport in the HTML header and adjust the screen size.

```
<meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-
scale=1, user-scalable=no">
```

- The Windows version does not support the viewport setting.

4.8 Creating war files

Applications must be in the war format to be deployed to the AR processing server. Therefore, web applications that you have created must be converted to this format.

4.8.1 Using the jar command

This section explains how to create war files using the jar command provided by Java.

1. Create the Web-INF directory and web.xml

   - Create the Web-INF directory directly under the context directory of the web application, and store web.xml that specifies the required settings.
   - To cache web resources, set mimetype used in web.xml to enable the cache configuration file.

      Example

     To use the appcache file:

     ```
     <mime-mapping>
       <extension>appcache</extension>
       <mime-type>text/cache-manifest</mime-type>
     </mime-mapping>
     ```

2. Create war files.

   a. Using the command prompt, move to the root directory where you developed the application.

   b. Use the jar command provided by Java to create a webapp.war file.
   The directory structure when the jar command is deployed depends on the current directory at the time the command is run, so check the directory before running the command.
   Refer to the Java manual for details on the jar command.

    Example

   Example of executing the jar command:

   ```
   "C:\Program Files\Fujitsu\common\jdk7\bin\jar" cvf ../webapp.war *
   ```

Chapter 5 Developing the overlay application (native application)

This chapter explains how to develop the native application, which is a component of the overlay application.

5.1 Overview

The development resources provided by Interstage AR Processing Server enable you to develop native applications that:

- Perform client authoring.
- Use the overlay application development tool on smart devices to run the sample application deployed on a web server and the overlay application itself.
- Use the AR development project to embed this product's features into applications developed by the system developer.

5.1.1 Development resources provided by Interstage AR Processing Server

Interstage AR Processing Server provides the following resources for developing the native application:

W

- Resources provided for the Android version
  - *installDir*\mobileclient\android\ARClientAuthoring.apk
    Client authoring tool [executable file, apk format]
  - *installDir*\mobileclient\android\ARClientDev.zipAR
    AR development project [Eclipse development project, zip format]
  - *installDir*\mobileclient\android\ARClientDev.apk
    Overlay application (web application) development tool [executable file, apk format]
- Resources provided for the iOS version
  - *installDir*\mobileclient\ios\ARClientAuthoring.zip
    Client authoring tool development project [Xcode development project, zip format]
  - *installDir*\mobileclient\ios\ARClientDev.zip
    AR development project [Xcode development project, zip format]
- Resources provided for the Windows version
  - *installDir*\mobileclient\Windows\ARMobileClientSetup.msi
    Client authoring tool and overlay application (web application) development tool [installer, msi format]
  - *installDir*\mobileclient\Windows\ARClientDev.zip
    AR development project [Visual Studio 2013 development project, zip format]
- Common
  - *installDir*\lib\js*
    JavaScript library
  - *installDir*\samples\quickstart*
    Sample application (web application)

L

- Resources provided for the Android version
  - /opt/FJSVar/mobileclient/android/ARClientAuthoring.apk
    Client authoring tool [executable file, apk format]
  - /opt/FJSVar/mobileclient/android/ARClientDev.zip
    AR development project [Eclipse development project, zip format]
  - /opt/FJSVar/mobileclient/android/ARClientDev.apk
    Overlay application (web application) development tool [executable file, apk format]
- Resources provided for the iOS version

- /opt/FJSVar/mobileclient/ios/ARClientAuthoring.zip
  Development project for the client authoring tool [Xcode development project, zip format]
- /opt/FJSVar/mobileclient/ios/ARClientDev.zip
  AR development project [Xcode development project, zip format]

- Resources provided for the Windows version
  - /opt/FJSVar/mobileclient/Windows/ ARMobileClientSetup.msi
    Client authoring tool and overlay application (web application) development tool [installer, msi format]
  - /opt/FJSVar/mobileclient/Windows/ARClientDev.zip
    AR development project [Visual Studio 2013 development project, zip format]
- Common
  - /opt/FJSVar/lib/js/*
    JavaScript library
  - /opt/FJSVar/samples/quickstart/*
    Sample application (web application)

The executable file for the client authoring tool (iOS) is not offered as a product, because Apple Inc. does not permit distribution of executable files. To execute the client authoring tool on iOS, the client company must enter into a contract with Apple Inc. and build the client authoring tool before it can be used.

5.1.2 Development flow

5.1.2.1 Building the development environment

You must prepare the following environments for developing the native application to be used with Interstage AR Processing Server.

To prepare the development environment, install and set up each product in accordance with the manual provided with that product.

Android

To develop an Android version of the native application, prepare the development environment by following the procedure below:

1. Prepare the following environments:
   - Android SDK 4.4.2 (API 19)
   - Eclipse IDE
   - JDK 6
   - Android Development Tools (ADT) plug-in
   - Extract ARClientDev.zip to the desired folder.
2. Import the project file to Eclipse.

iOS

To develop an iOS version of the native application and build a client authoring tool, prepare the development environment by following the procedure below:

1. Prepare the following environments:
   - Xcode 5.0 or later
2. Obtain a development certificate.

3. Extract the Xcode development project to the desired folder.
   - To develop the native application, extract the ARClientDev project.
   - To build the client authoring tool, extract the ARClientAuthoring project.
4. Import the project file to Xcode.
   - To develop the native application, click ARClientDev.xcodeproj and import it.
   - To build a client authoring tool, click ARClientAuthoring.xcodeproj and import it.

Windows

To develop a Windows version of the native application, prepare the development environment by following the procedure below:

1. Prepare the following environments:
   - Visual Studio(R) 2013
     One of the following editions is required:
     - Professional
     - Premium
     - Ultimate
2. Extract ARClientDev.zip to the desired folder.
3. Open the solution file (ARClientDev.sln) in Visual Studio.

5.1.2.2 Building

Use Eclipse or Xcode to build a project, and use Visual Studio to build a solution. Refer to the manual for your development environment for details on how to build. Refer to "5.2 Developing an Android native application", "5.3 Developing an iOS native application", and "5.4 Developing a Windows native application" for details on developing the native application.

5.1.2.3 Installing the developed application

Install the application that you built onto a smart device - refer to the *Operator's Guide* for details.

5.2 Developing an Android native application

5.2.1 Application type and supported operations

The native application that the system developer can develop calls the authoring screen and the overlay application activity.

Table 5.1 Application type and supported operations

Application type	Development language	Supported operations
Native application	Native language, such as Java and Objective-C	- Call the authoring activity, overlay application activity, and configuration activity (start the activities), and create a native application.
Overlay application	JavaScript	- Use JavaScript Render AR overlay content, communicate with the AR server, enable offline use, retrieve marker recognition information, etc. - Run user-defined web applications on the overlay application.

Figure 5.1 Image of native application operation

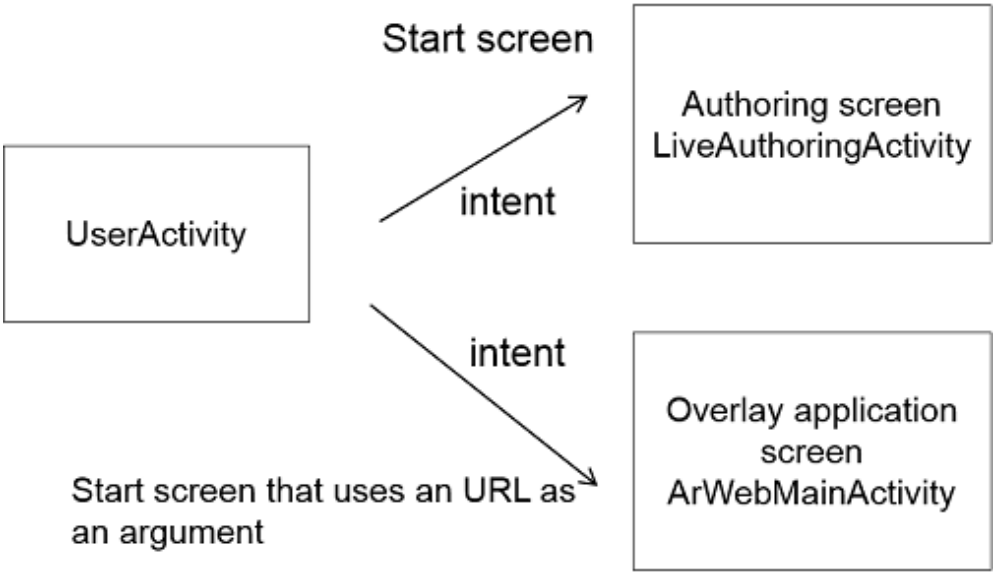
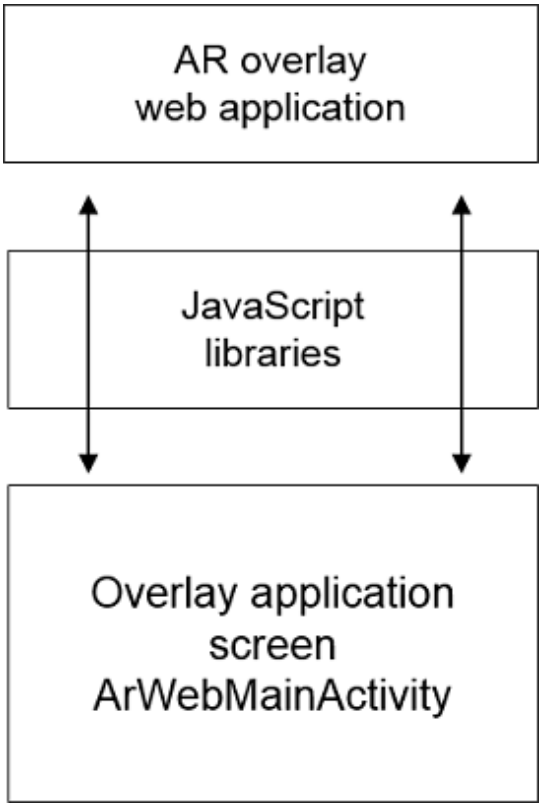


Figure 5.2 Image of overlay application operation



5.2.2 Configuring the project

This section explains the configuration of the ARClientDev project.

The root folder is ARClientDev - resources marked with a number sign (#) can be edited, but resources marked with an asterisk (*) are product files, and hence cannot be edited or deleted.

```
asssets

       authoring: Image folder used for product authoring *

       logutil.properties: Product log configuration file *
```

```
libs: Library folder
       armeabi-v7a
              libarengine.so: Product library (C++) *
       arclientlib.jar: Product library (Java) *
src: Sample source folder
       com.fujitsu.interstage.ar.mobileclient.android
              R.java: Product resource file *
       com.sample.clientdev
              MainMenuActivity.java    List menu screen #
res: Resources folder
       drawable[option]: drawable folder
              Product image file in the folder *
              aricon.png: Product sample icon #
       layout:  layout folder
              Product layout file in the folder *
       raw: raw folder
              Product raw file in the folder *
       values: value folder (common)
              Product value file in the folder *
              strings.xml: Character resource file *
       values-ja:  value folder (Japanese)
              Product value file in the folder *
              strings.xml: Character resource file #
AndroidManifest.xml: Android manifest file #
```

5.2.3 Starting activities

Using the development project (ARClientDev) as the base, develop the user native application by adding user activities.

The native application provides the following three activities:

- ArWebMainActivity
  Overlay application activity.
  It superimposes the overlay application onto the camera layer and the AR overlay layer. During startup, it loads the overlay application from the web server to the device according to the specified URL. Refer to "Chapter 4 Developing the overlay application (web application)" for details.
- LiveAuthoringActivity
  Client authoring activity.
  It is used for adding and editing AR overlay content. Refer to the *Operator's Guide* for details on how to use the client authoring tool.

- ArPreferenceActivity
  Configuration screen activity.
  It displays the configuration screen for retrieving application logs and setting up accounts. Refer to the *Operation Guide* for details.

By using "intent" to start the above activities from the native application developed by the system developer, you can create an executable file (apk) that combines developed activities and activities provided by Interstage AR Processing Server. Select the activities to be used and embed them into the executable file:

- To use the overlay application only:
  Embed ArWebMainActivity and ArPreferenceActivity into the application.
- To use client authoring only:
  Embed LiveAuthoringActivity and ArPreferenceActivity into the application.
- To use both:
  Embed all activities into the application.

Refer to the sample com.sample.clientdev.MainMenuActivity.java for an example of how to call an activity.

Table 5.2 Intent specifications of the overlay application activity

Class name	com.fujitsu.interstage.ar.mobileclient.android.base.web.ArWebMainActivity
Extra	- Activation URL key: "extraskey_widget_url" value: URL string to be activated

Table 5.3 Intent specifications of the client authoring activity

Class name	com.fujitsu.interstage.ar.mobileclient.android.authoring.ui.LiveAuthoringActivity
Extra	None

Table 5.4 Intent specifications of the configuration screen activity

Class name	com.fujitsu.interstage.ar.mobileclient.android.base.pref.ArPreferenceAcitity
Extra	None

Example

To start the overlay application activity:

```
//Set the overlay application activity in an intent.
String _activity = "com.fujitsu.interstage.ar.mobileclient.android.base.web.ArWebMainActivity";
Intent intent = new Intent();
intent.setClassName(MainMenuActivity.this, _activity);
//Set the URL to be activated in the intent.
// key:  extraskey_widget_url
// value:  URL
intent.putExtra("extraskey_widget_url", "https://www.fujitsu.com/index.html");
//Start the activity.
startActivity(intent);
```

To start the client authoring activity:

```
//Set the client authoring activity in an intent.
String _activity =
"com.fujitsu.interstage.ar.mobileclient.android.authoring.ui.LiveAuthoringActivity";
Intent intent = new Intent();
intent.setClassName(MainMenuActivity.this, _activity);
//Start the activity.
startActivity(intent);
```

To start the configuration screen activity:

```
//Set the configuration screen activity in an intent.
String _activity = "com.fujitsu.interstage.ar.mobileclient.android.base.pref.ArPreferenceAcitity";
Intent intent = new Intent();
intent.setClassName(MainMenuActivity.this, _activity);
//Start the activity.
startActivity(intent);
```

To set the start screen for a user overlay application

MainMenuActivity uses a button click event as a trigger to start ArWebMainActivity in an intent. To directly start the user overlay application after startup, start ArWebMainActivity using the onCreate method.

Note

For security reasons, the above activities are configured as private from external intents, by setting android:exported="false" in AndroidManifest.xml.

5.2.4 Understanding the orientation of the activity screens

This section explains the orientation of each activity screen.

- ArWebMainActivity
  The orientation is set to portrait or landscape depending on the orientation of the smart device during startup, and subsequently remains fixed.
- LiveAuthoringActivity
  The orientation is set to portrait mode if the smart device is a smart phone, or to landscape mode if it is a tablet.
- ArPreferenceActivity
  The orientation varies between portrait and landscape modes depending on the orientation of the smart device.

5.2.5 Editing AndroidManifest.xml

To develop an application using a development project, edit AndroidManifest.xml. Do not change the descriptions used on the product side, otherwise a compilation error or abnormal operation may occur.

As a rule, you cannot change the Activity/ContentProvider/BroadcastReceiver or permission settings used by the product.

You can add settings to be used by user applications:

- Add a user application activity
- Add intent-filter to a user application activity
- Add user-permission to be used by a user application

Example

Refer to the comments in the source code for details on whether the source code can be edited.

```
<!-- Application settings for the product. Cannot be edited. Start-->
    <application
        android:name="com.fujitsu.interstage.ar.mobileclient.android.base.ArMainApplication"
        android:debuggable="false"
        android:icon="@drawable/aricon"
        android:label="@string/app_name_dev"
        android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
    <!-- Application settings for the product. Cannot be edited.  End-->

        <!-- Activity for the sample. Can be edited.  Start-->
```

```
        <activity
            android:name=".MainMenuActivity"
            android:configChanges="keyboardHidden|orientation"
            android:label="@string/app_name_dev"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Activity for the sample. Can be edited.  End-->
```

To change the startup activity of the launcher

The launcher is configured with intent-filter for setting MainMenuActivity. To change the startup activity, change the setting destination activity in the following intent-filter settings.

 Example

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

5.2.6 Editing res/values

If editing the files (strings.xml, arrays.xml, attrs.xml, colors.xml, dimens.xml, styles.xml) under the res/values folder, do not delete or edit content already specified in the product. Descriptions can be added for a user application, as long as the names do not conflict with existing attribute names.

 Example

Example of editing res/values/string.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<resources>
    <!-- Strings set in the product. Start -->
    <string name="app_name">InterstageAR</string>
    <string name="app_name_authoring">InterstageAR Authoring</string>
    <string name="app_name_dev">InterstageAR Dev</string>
   .....
    <!-- Strings set in the product. End -->
    <!-- Definitions can be added as follows -->
    <string name="user_add_string1">user-defined string 1 </string>
    <string name="user_add_string2">user-defined string 1 </string>
    <!-- You cannot add a name already set in the product, as the one below, because a name conflict
will occur -->
    <string name="app_name">user application</string>
</resources>
```

5.2.7 Changing the startup icon

To change the launcher icon, replace the icon aricon.png in the following folders with the desired image:

- res/drawable-xdpi/aricon.png
- res/drawable-ldpi/aricon.png

- res/drawable-mdpi/aricon.png

5.2.8 Changing the application name

To change the application name, change the files below:

- Application name to be displayed when the language setting is English
  res/values/strings.xml

```
<string name="app_name_dev">InterstageAR Dev</string>
```

- Application name to be displayed when the language setting is Japanese
  res/values-ja/strings.xml

```
<string name="app_name_dev">InterstageAR Dev</string>
```

5.2.9 Changing the package name

Follow the procedure below to change the package name in a user application:

1. Change the package name using AndroidTools
   - Right-click the development project (ARClientDev), and select **Android Tools**, and then **Rename Application Package**.
   - Type the new package name and click **OK** (example: change to com.test).
   - A list of the files to be changed is displayed. Confirm them and execute rename.
2. Change the ContentProvider description in AndroidManifest.xml
   Change "com.sample.clientdev" specified in
   android:authorities="com.sample.clientdev.data.httpdata.httpdataprovider" to the desired package name.

5.3 Developing an iOS native application

When developing an iOS native application, the system developer should use the ARClientDev project.

Note

Do not use the ARClientAuthoring project for any purpose other than to build a client authoring tool.

5.3.1 Project

This section explains the configuration of the ARClientDev project. Resources marked with a number sign (#) can be edited, but resources marked with an asterisk (*) are product files, and hence cannot be edited or deleted.

```
ARClientDev: Root folder

       Resources: Resources folder

              drawable: Image data folder

                     (Product image file) *

       en.lproj

       Localizable.strings: English text file #

       ja.lproj
```

```
Localizable.strings: Japanese text file #

values      values folder

        (Product values file) *

ARClientLib.framework: Product library folder

        (Product library file in the folder) *

opencv2.framework: Library folder used by the product

        (Library file used by the product in the folder) *

ARClientDev: Sample source folder

       en.lproj

       InfoPlist.strings: English text file #

       ja.lproj

       InfoPlist.strings: Japanese text file #

       IARMainMenuViewController.h: List menu screen (.h) #

       IARMainMenuViewController.m: List menu screen (.m) #

        (Sample source file in the folder) #

Icon_114.png: Icon image #

Icon_144.png: Icon image #

Icon_57.png: Icon image #

Icon_72.png: Icon image #
```

5.3.2 Starting UIViewController

Using the development project (ARClientDev) as the base, develop the user native application by adding the user UIViewController.

This product contains the following three UIViewControllers:

- IARWebMainViewController
  Overlay application UIViewController. It superimposes the overlay application onto the camera layer and the AR overlay layer. During startup, it loads the overlay application from the web server to the device according to the specified URL. Refer to "Chapter 4 Developing the overlay application (web application)" for details.
- IARLiveAuthoringViewController
  Client authoring UIViewController. It is used for adding and editing AR overlay content. Refer to the *Operator's Guide* for details on how to use the client authoring tool.
- IARSettingViewController
  Configuration screen UIViewController. It displays the configuration screen for retrieving application logs and setting up accounts. Refer to the *Operator's Guide* for details.

You can start the above UIViewControllers from the native application. You can therefore create an executable file (app) that combines developed UIViewControllers and the UIViewControllers provided by the product. Select the UIViewControllers to be used and embed them into the executable file:

- To use the overlay application only:
  Embed IARWebMainViewController and IARSettingViewController into the application.
- To use client authoring only:

Embed IARLiveAuthoringViewController and IARSettingViewController into the application.

- To use both:
  Embed all UIViewControllers into the application.

Refer to ARClientDev/IARMainMenuViewController.m for an example of how to call a UIViewController.

Starting UIViewController

IARMainMenuViewController uses a button click event as a trigger to start IARWebMainViewController. To directly start the user overlay application after startup, start IARWebMainViewController using the viewDidLoad method.

Example

Overlay application UIViewController

- Example of calling IARWebMainViewController:

```
//Set the URL to be activated.
NSString *sampleUrl = @"https://www.fujitsu.com/index.html";
// Create UIViewController.
[IARMainController getInstance].mainViewCtrl = [[IARWebMainViewController alloc] initWithUrl:
sampleUrl inputClassName:NSStringFromClass([IARMainMenuViewController class])];
//Start UIViewController.
self.window.rootViewController = [IARMainController getInstance].mainViewCtrl;
```

- Example of calling IARLiveAuthoringViewController:

```
// Create UIViewController.
[IARMainController getInstance].mainViewCtrl = [[IARLiveAuthoringViewController alloc]
initWithUrl:  sampleUrl inputClassName:NSStringFromClass([IARMainMenuViewController class])];
//Start UIViewController.
self.view.window.rootViewController = [IARMainController getInstance].mainViewCtrl;
```

- Example of calling IARSettingViewController:

```
// Create UIViewController.
id sampleViewController = [[IARSettingViewController alloc] init];
//Start UIViewController.
[self presentModalViewController:  sampleViewController animated:NO];
```

To change the startup UIViewController of the launcher

The launcher is configured to display IARMainMenuViewController during startup. To change the startup UIViewController, set the UIViewController you want to start in self.window.rootViewController.

Example

- ARClientDev/AppDelegate.mm
  didFinishLaunchingWithOptions

```
    self.window.rootViewController = [[IARMainMenuViewController alloc] init];
```

5.3.3 Orientation of the UIViewController screens

The screen orientation is set as follows:

- iPhone: Portrait
- iPad: Landscape

Note

If you change the screen orientation setting, IARLiveAuthoringViewController and IARWebViewMainViewController cannot be displayed properly and may be terminated abnormally

5.3.4 Editing ARClientDev-info.plist and build settings

To develop the native application, edit info.plist. Do not change the descriptions used on the product side, otherwise a compilation error or abnormal operation may occur.

You must not change the settings for the following:

- UISupportedInterfaceOrientations and UIFileSharingEnabled in ARClientDev-info.plist
- iOS Deployment Target (iOS 5.0) in build settings

Example

Refer to the comments in the following source code for details on whether the source can be edited:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CFBundleDevelopmentRegion</key>
    <string>ja_JP</string>
    <key>CFBundleDisplayName</key>
    <string>InterstageAR</string>
    <key>CFBundleExecutable</key>
    <string>${EXECUTABLE_NAME}</string>
    <key>CFBundleIcons</key>
    <dict>
        <key>CFBundlePrimaryIcon</key>
        <dict>
            <key>CFBundleIconFiles</key>
            <array>
                <string>Icon_57.png</string>
                <string>Icon_114.png</string>
                <string>Icon_72.png</string>
                <string>Icon_144.png</string>
                <string>Icon_57.png</string>
                <string>Icon_114.png</string>
                <string>Icon_72.png</string>
                <string>Icon_144.png</string>
            </array>
            <key>UIPrerenderedIcon</key>
            <true/>
        </dict>
    </dict>
    <key>CFBundleIdentifier</key>
    <string>com.fujitsu.${PRODUCT_NAME:rfc1034identifier}</string>
    <key>CFBundleInfoDictionaryVersion</key>
    <string>6.0</string>
    <key>CFBundleName</key>
    <string>${PRODUCT_NAME}</string>
    <key>CFBundlePackageType</key>
    <string>APPL</string>
    <key>CFBundleShortVersionString</key>
    <string>1.0</string>
    <key>CFBundleSignature</key>
    <string>????</string>
```

```
    <key>CFBundleVersion</key>
    <string>1.0</string>
<!-- Import/export feature for the product. Cannot be edited.  Start-->
    <key>UIFileSharingEnabled</key>
<true/>
<!-- Import/export feature for the product. Cannot be edited.  End-->
    <key>UIPrerenderedIcon</key>
    <true/>
    <key>UIRequiredDeviceCapabilities</key>
    <array>
        <string>armv7</string>
    </array>
<!-- SupportedInterfaceOrientations for the product. Cannot be edited.  Start-->
    <key>UISupportedInterfaceOrientations</key>
    <array>
        <string>UIInterfaceOrientationPortrait</string>
    </array>
    <key>UISupportedInterfaceOrientations~ipad</key>
    <array>
        <string>UIInterfaceOrientationLandscapeRight</string>
    </array>
<!-- SupportedInterfaceOrientations for the product. Cannot be edited.  End-->
</dict>
</plist>
```

5.3.5 Editing Localizable.strings, InfoPlist.strings, and arrays.plist

If editing Localizable.strings and InfoPlist.strings, do not delete or edit content already specified in the product. Descriptions can be added for a user application, as long as the names do not conflict with existing attribute names.

Example

- Example of editing Localizable.strings

```
    <!-- String set in the product. Start -->
    "app_name"= "InterstageAR";
    "app_name_authoring" = "InterstageAR Authoring";
   "app_name_dev" = "InterstageAR Dev";
   .....
    <!-- String set in the product. End -->
    <!-- You can add definitions as shown below. -->
    "user_add_string1" = "user-defined string 1";
    "user_add_string2" = "user-defined string 1";
```

- Example of editing InfoPlist.strings

```
<!-- Application name -->
 CFBundleDisplayName = " InterstageAR";
```

5.3.6 Changing the startup icon

To change the launcher icon, replace the icon in the following folders with an image of with the same dimensions:

- ARClientDev/Icon_57.png
- ARClientDev/Icon_72.png
- ARClientDev/Icon_114.png
- ARClientDev/Icon_144.png

5.3.7 Changing the application name

To change the application name, change the files below:

Example

- Application name to be displayed when the language setting is English
  ARClientDev/en.lproj/InfoPlist.strings

```
CFBundleDisplayName = " InterstageAR";
```

- Application name to be displayed when the language setting is Japanese
  ARClientDev/ja.lproj/InfoPlist.strings

```
CFBundleDisplayName = " InterstageAR";
```

5.4 Developing a Windows native application

5.4.1 Solution

This section explains the configuration of the ARClientDev solution. Resources marked with a number sign (#) can be edited, but resources marked with an asterisk (*) are product files, and hence cannot be edited or deleted.

The root folder is ARClientDev.

```
ARClientDev: Root folder

       ARClientDev.sln       Solution file #

       ARClientDev           Project folder

          ARClientDev.csproj    Project file #

          App.xaml              Application definition file  #

          MainMenuPage.xaml    Main menu page #

          MainWindow.xaml          Call main menu screen *

          Styles.xaml           Application UI settings information #

          Properties: Product properties folder #

          ReferenceAssemblies: Product library folder *
```

5.4.2 Calling control

Using the development project (ARClientDev) as the base, develop the user native application by embedding Page control provided by the product.

The native application provides the following controls:

- ArWebMainPage
  Page control of the AR overlay application. It superimposes the overlay application onto the camera layer and the AR overlay layer. During startup, it loads the overlay application from the web server to the device according to the specified URL. Refer to "Chapter 4 Developing the overlay application (web application)" for details.
- SelectScenarioPage
  Client authoring control. It adds and edits AR overlay content. Refer to the *Operator's Guide* for details on how to use the client authoring tool.

- ArPreferenceWindow
  Settings screen control. It displays the settings screen for configuring such settings as the application logs, accounts, and camera. Refer to the *Operator's Guide* for details.

By calling the above controls from the native application developed by the system developer, you can create an application that combines the features provided by Interstage AR Processing Server.

- To use the overlay application only:
  Embed ArWebMainPage and ArPreferenceWindow into the application.
- To use client authoring only:
  Embed SelectScenarioPage and ArPreferenceWindow into the application.
- To use both:
  Embed all Page classes and Window classes into the application. Refer to the sample MainMenuPage.xaml.cs for an example of how to call controls.

Table 5.5 Calling an AR overlay application Page class constructor

Namespace	Fujitsu.Interstage.AR.Mobileclient.Windows.Base.Web
Class name	ArWebMainPage
Parameter	URL string to be activated string url

Table 5.6 Calling a client authoring Page class constructor

Namespace	Fujitsu.Interstage.AR.Mobileclient.Windows.Authoring.UI
Class name	SelectScenarioPage
Parameter	None

Table 5.7 Calling a settings screen Window class constructor

Namespace	Fujitsu.Interstage.AR.Mobileclient.Windows.Base.Pref
Class name	ArPreferenceWindow
Parameter	None

Example

- To start the AR overlay application Page class:

```
// Set the AR overlay application URL.
string url = "https://www.fujitsu.com/index.html";

// Use the NavigationService class to call the Page class.
NavigationService.Navigate(new ArWebMainPage(url));
```

- To start the client authoring Page class:

```
// Use the NavigationService class to call the Page class.
NavigationService.Navigate(new SelectScenarioPage());
```

- To start the settings screen Window class:

```
// Use the NavigationService class to call the Window class.
NavigationService.Navigate(new ArPreferenceWindow());
```

5.4.3 Screen orientation

The orientation of each screen varies between portrait and landscape modes depending on the orientation of the smart device. However, the screen orientation becomes fixed while the camera image is stopped during operations such as the following:

- When authoring is being used to edit an AR overlay definition

- When the native camera has been stopped by stopCameraView() of the native application

Note

Screen orientation becomes fixed when the AR overlay application window is selected. If a value other than the AR overlay application is selected, the screen orientation changes depending on the orientation of the smart device, and the displayed overlay content may be displayed in the wrong place.

5.4.4 Changing the startup icon

To change the icon to be embedded in an application, replace the icon in the following folder with the desired image:

- ARClientDev\product.ico

5.4.5 Changing the application name

To change the application name, change the files below:

1. Open the properties for the **ARClientDev** project.
2. Click the **Application** tab, and specify an application name in **Assembly Name**.

5.4.6 Changing the default namespace

To change the default namespace for an application, change the application name as explained below:

1. Open the properties for the **ARClientDev** project.
2. Click the **Application** tab, and set a name for **Default namespace**.

5.4.7 Entering data to text boxes

When you enter a string in a text box, the Windows touch keyboard is not automatically displayed. Display and use the touch keyboard as required.

5.4.8 Distributing applications

When distributing an application, deploy all libraries contained in the product library folder to the same folder as the application to be used.

This product also uses the libraries listed below. Install these libraries to the deployment environment or to the environment where the product installer will be deployed.

- Microsoft Visual C++ 2010 Redistributable Package
- Microsoft Visual C++ 2012 Redistributable Package

Chapter 6 Developing the system integration application

This chapter explains how to develop the system integration application.

6.1 Overview

The system integration application is created for using data from an existing business system in the overlay application, or for reflecting data input using the overlay application to an existing business system.

The REST API published in Interstage AR Processing Server enables you to retrieve, register, and delete AR overlay definition information managed by the server.

You can create the system integration application using any programming language (such as Java and C). You can even deploy the system integration application as a web application, or create it as a standalone application that is to be run periodically using a scheduler such as cron. The developer must decide these details taking into account the purpose of the system.

Point

We cannot guarantee correct operation of system integration applications deployed to the server provided by this product. Please prepare a separate application server when deploying a system integration application.

6.1.1 Development environment

Select the programming language to be used for development, taking into account the purpose of the system integration application, the application server on which it is to be deployed, and integration with the core business system. Also prepare the development environment required for your selected programming language.

6.1.2 Development flow

This section explains how to develop the system integration application.

1. Program the system integration application.
   Code the system integration application. It is divided into "data reflection from the core business system to the AR processing server" and "data reflection from the AR processing server to the core business system".
2. Compile and package the system integration application.
   Compile the system integration application using the development environment, store it in a war file, and package it.

   Point

   If the system integration application was not created using Java, then it must be compiled and linked as required by the programming language used.
3. Deploy the system integration application
   Deploy the packaged system integration application to the application server. Refer to the manual for your application server for details on how to deploy the application. If you create the system integration application in any format other than a web application, follow the appropriate procedure to enable the system integration application to be used.

6.2 Programming

The system integration application consists of one or both of the following: data reflection from the core business system to the AR processing server, and data reflection from the AR processing server to the core business system.

6.2.1 Reflecting data from the core business system to AR processing server

Create the following process to reflect data from the core business system to the AR processing server:

1. Retrieve data from the core business system.

2. Convert data for AR processing server.
3. Send data to AR processing server.

Use the interface provided by the core business system to retrieve data from it -if it uses an interface that sends data to the system integration application, then you must create a mechanism for the system integration application to receive the data.

Data conversion converts the data format provided by the interface of the core business system to a format that the AR processing server can receive.

A simple method is to assign the parameter names and parameter values of data in the core business system to the respective elements in the AR overlay definition. Another method is to design a data schema and convert data in accordance with it (the data schema must be created taking into account the purpose of the system).

During data conversion, use the REST API to retrieve other data from the core business system and send it to the AR processing server. You cannot register QType or QAttribute using the REST API. Therefore, you must build a user-defined table before storing the retrieved data in the AR processing server.

Item	Registration (POST)	Identity via primary key API (GET)	Search (GET)	Delete (DELETE)
QType	N	Y	Y	N
QAttribute	N	Y	Y	N
QEntity	Y	Y	Y	Y
QValue	Y	Y	Y	Y
Quad	Y	N	Y	N

Example

The processing sequence if template information containing AR content position information for each job is stored in the AR processing server beforehand is as follows:

1. Retrieve job instruction data from the core business system.
2. Retrieve template information containing job instructions from AR processing server.
3. Browse the template information containing job instructions and append position information about the AR object to the job instruction data.
4. Reflect the job instruction data, to which position information for AR overlay content was appended, to the AR processing server.

6.2.2 Reflecting data from the AR processing server to the core business system

Create the following process to reflect data from the AR processing server to the core business system:

1. Retrieve data from the AR processing server.
   Use the REST API (get) to retrieve data from the AR processing server.
2. Convert and reflect data to the core business system.
   Use the interface provided by the core business system to reflect data to it.

Example

For example, the processing sequence for reflecting job results entered by a worker on-site using a smart device to a core business system is as follows:

1. The AR business application stores the job execution results in the AR processing server.
2. The AR business application reports the job completion event to the system integration application.
3. The system integration application retrieves the job results data from the AR processing server.

4. The system integration application converts JSON format job results data for the core business system.
5. The system integration application reflects the converted job results data to the core business system.

6.3 Notes on creating the system integration application

There is no feature for automatically calling APIs when data in the AR processing server is updated. Therefore, a mechanism is required for calling a process to reflect data from the AR processing server to the core business system.

Example

Example mechanisms:

- The AR business application reports data update to the system integration application when data in the AR processing server is updated.
- The system integration application periodically checks if data in the AR processing server has been updated.

Chapter 7 Sample application (AR marker)

This chapter explains the basic sample application (AR marker) that uses AR markers provided by Interstage AR Processing Server, focusing on the anticipated job content, the composition of the sample application, and how to run it.

7.1 Overview of the sample application (AR marker)

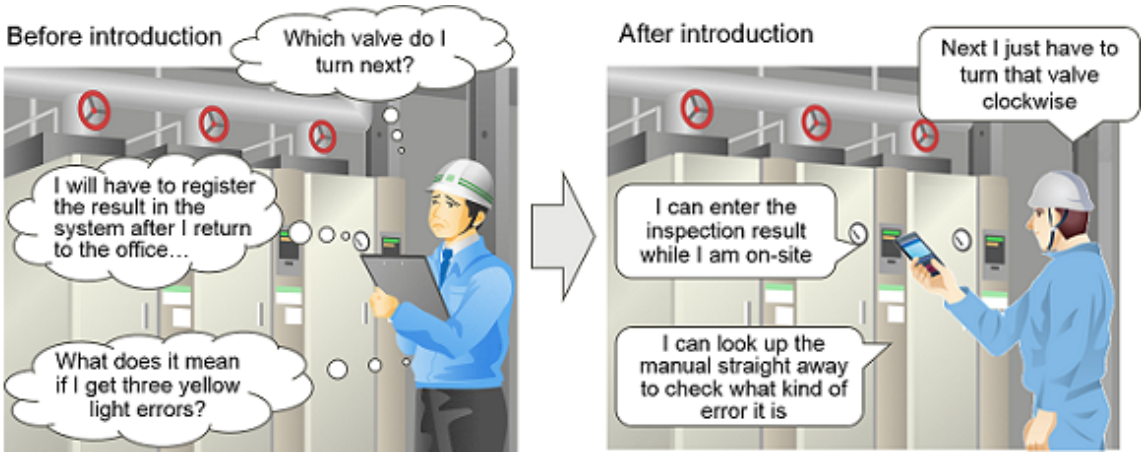
7.1.1 Job content

Before the introduction of Interstage AR Processing Server, a worker conducting an equipment inspection would check the job content based on the job inspection sheet and fill out the inspection result. If an unexpected problem occurred, the worker would note down or remember the model number of the equipment and look for the corresponding manual at the office.

The sample application (AR marker) has the following features to support equipment inspection jobs:

1. Provides instructions on operating valves.
   Previously, the worker had to check the work sequence while comparing the actual object with the inspection sheet, but Interstage AR Processing Server superimposes the inspection sheet onto a captured image of the real-world object.
2. Displays the equipment manual.
   Previously, if a problem occurred, the worker had to search for the corresponding manual at the office, but Interstage AR Processing Server enables the manual to be checked immediately on-site.
3. Inputs the inspection result.
   Previously, the worker registered the written inspection result after returning to the office, but Interstage AR Processing Server enables the result to be registered immediately.

Figure 7.1 Equipment inspection job before and after the introduction of Interstage AR Processing Server



7.1.2 Composition of the sample application (AR marker)

The sample application (AR marker) provided by Interstage AR Processing Server consists of the files listed below.

Point

The sample application (AR marker) provided by Interstage AR Processing Server is stored in the following location:

W *installDir*\samples\quickstart\

L /opt/FJSVar/samples/quickstart/

```
background.pdf: Background image used in the sample application (AR marker)

file: Directory with files used for overlaying in the sample application
```

```
      INPUT.png

      MANUAL.pdf

      MANUALPOP.png

      MOVIE.mp4

      MOVIEPOP.png

      PICTURE.jpg

      THUMBNAIL.jpg
webapp: Sample application (AR marker) directory
      ar: JavaScript library storage directory
         ar.js: JavaScript library file
      apl_sample: JavaScript directory for running the sample application (AR marker)
         apl.js: JavaScript file for running the sample application (AR marker)
      WEB_INF: Settings file directory for deploying the sample application (AR marker)
         web.xml: Settings file
      index.html: Start screen HTML file
      perwork.html: Work preparation screen HTML
      work.html: Work screen HTML
      comment.html: Comment input screen HTML file
      gyoumu.css: Sample application (AR marker) CSS file
      work.appcache: Application cache settings file
```

7.1.3 Features used by the sample application (AR marker)

The following table indicates which features provided by the JavaScript libraries are used by the sample application (AR marker).

Library	Features	Used by the sample application (AR marker)
Camera control	Start/stop native camera	Y
	Register/remove marker detection	Y
	Retrieve ID and rectangular coordinate of detected marker	N
Log	Output log	Y
OS control	Report click coordinate to native	Y
	Display specified URL content	N
	Check operation mode	Y
	Call settings screen	N
	Recognition mode settings	Y
AR rendering	Operate AR content: Retrieve	N

Library	Features	Used by the sample application (AR marker)
	Operate AR content: Add	Y
	Operate AR content: Remove	Y
	Operate AR content: Remove all	Y
Data management control	Set AR processing server information	Y
	Set use of offline storage	Y
	Communicate with the AR processing server: Retrieve	Y
	Communicate with AR processing server: Register	Y
	Communicate with AR processing server: Remove	N
	Clear offline storage	Y

7.2 Preparing the sample application (AR marker)

This section explains how to install and configure the server for running the sample application (AR marker).

Point

The sample application (AR marker) displays PDF files. Ensure beforehand that the PDF files can be viewed on a smart device.

7.2.1 Installing and configuring the server

Install and configure Interstage AR Processing Server - refer to the *Installation Guide* for details.

7.2.2 Deploying the web application

Create a sample web application (AR marker) and deploy it so that it can be accessed from a smart device.

This example explains how to deploy the sample web application to the AR processing server. If deploying the sample web application to a model C or D configuration or other non-AR processing server environment, follow the procedure for deploying your application server.

1. Use the command prompt to redirect to the webapp directory.
2. Use jar provided by JDK to create a webapp.war file. The directory structure when jar is deployed depends on the current directory, so redirect to the webapp directory before running the command.
   Refer to the JDK manual provided by Oracle Corporation for details on jar.

   Example

   ```
   "C:\Program Files\Interstage\ISAR\components\STFISAR_HJDK7\jdk7\bin\jar.exe" -cvf ..\webapp.war
   *
   ```

3. Execute arsvadmin to deploy the webapp.war file.

   Example

   Deploying to the AR processing server

   Execute arsvadmin to deploy the application:

   ```
   "installDir\bin\arsvadmin" deploy ..\webapp.war
   ```

Undeploying from the AR processing server

To cancel deployment after the sample is run, execute the following command:

```
"installDir\bin\arsvadmin" undeploy webapp
```

See

Refer to the *Operator's Guide* for details on deployment.

7.2.3 Installing the native application

Install the sample native application provided by Interstage AR Processing Server to a smart device.

The Android version of the native application is stored in the following location:

W *installDir*\mobileclient\android\ARClientDev.apk

L /opt/FJSVar/mobileclient/android/ARClientDev.apk

Example

Installing the Android version of the native application via a USB connection

The example below installs the application on the smart device by connecting to a PC via USB (the USB connection method varies between smart devices - refer to the smart device manual for details).

1. Copy the sample native application to a PC that can be connected via USB to a smart device.
2. Select **Applications from unknown source** on the smart device.
   - On the **Settings** screen, tap **Security**, and then select **Unknown sources**.
3. Connect the smart device to the PC using a USB cable. If the PC does not recognize the driver, then install it.
4. On the PC, select the smart device using Windows Explorer - if the PC does not recognize the smart device, refer to the smart device manual.
5. Copy ARClientDev.apk to the desired location in the smart device.
6. Close the Windows Explorer.
7. On the PC taskbar, select **Safely remove hardware and eject media** and release the mounted device.
8. Start the application with the file management feature and select ARClientDev.apk. If there is no application with a file management feature, then install one.
9. The installation screen will be displayed. Start the installation.
10. Upon completion, on the **Settings** screen, **Security** menu, clear **Unknown sources**.

7.2.4 Configuring the native application

Configure the native application. Refer to the *Installation Guide* for details.

Point

In the AR server account settings explained in the *Installation Guide*, set the AR processing server account used in the sample application.

7.3 Managing data of the sample application (AR marker)

Use the data management console to register data in the system. Follow the procedure below to register data:

1. Assign the AR marker
   Assign the AR marker to be used in the sample application (AR marker).
2. Register a scenario
   Register data for the scenario used in the sample application (AR marker).
3. Register scenes
   Register the scenes used in the sample application (AR marker).
4. Register files
   Register the image files to be used in the sample application (AR marker).
5. Authoring
   Register AR overlay definitions to be used in the sample.
6. Create a user-defined table
   Create a user-defined table to be used in the sample application (AR marker).

See

Refer to the *Operator's Guide* for details on how to use the data management console.

Note

- If you enter different values for the data to be used by the sample application (AR marker), the application may not operate properly. Enter the data exactly as described in this chapter.
- You can change the scenario ID to be used by changing the value of Apl.useScenarioId in apl.js.
- Do not change the scene IDs or marker IDs.
- The sample application (AR marker) does not use the data entered for assigning a marker, so it runs even if you do not enter any data.
- AR overlay definition names are used for embedding user-defined data, so a null check is performed. If you leave the names blank, nothing is displayed. You should therefore always enter names.

7.3.1 Assigning the AR marker

Use the data management console to edit the AR marker with ID1, download it, and then print it.

Table 7.1 Content to be entered for the AR marker (AR marker ID1)

Item	Content to be entered
AR marker name	Job sample_Equipment A
Location	Basement 1, Kawasaki Plant
State	Change to **Used**.

Point

Alias is optional, so you need not enter it to the sample application (AR marker).

7.3.2 Registering a scenario

Use the data management console to register the following scenario data.

Table 7.2 Content to be entered for the scenario (inspection work)

Item	Content to be entered
Scenario ID	1
Scenario name	Inspection work

Item	Content to be entered
Description	Job sample_Inspection work scenario

7.3.3 Registering scenes

Use the data management console to register the following scenes for the **Inspection work** scenario:

Table 7.3 Scene list in inspection work

Item	Content to be entered			
Scene ID	1	2	3	4
Scene name	Operate valves	Check interior	Check value	Enter temperature
Description	Turn the valve clockwise.	Open the door and check the interior.	Check the monitor value.	Enter the temperature.

7.3.4 Registering files

Use the data management console to register the following files in the AR processing server. These files are used for authoring, so record the URLs of registered files.

Table 7.4 Files to be registered

File name	MIME type
INPUT.png	image/png
MANUAL.pdf	application/pdf
MANUALPOP.png	image/png
MOVIE.mp4	video/mp4
MOVIEPOP.png	image/png
PICTURE.jpg	image/jpeg
THUMBNAIL.jpg	image/jpeg

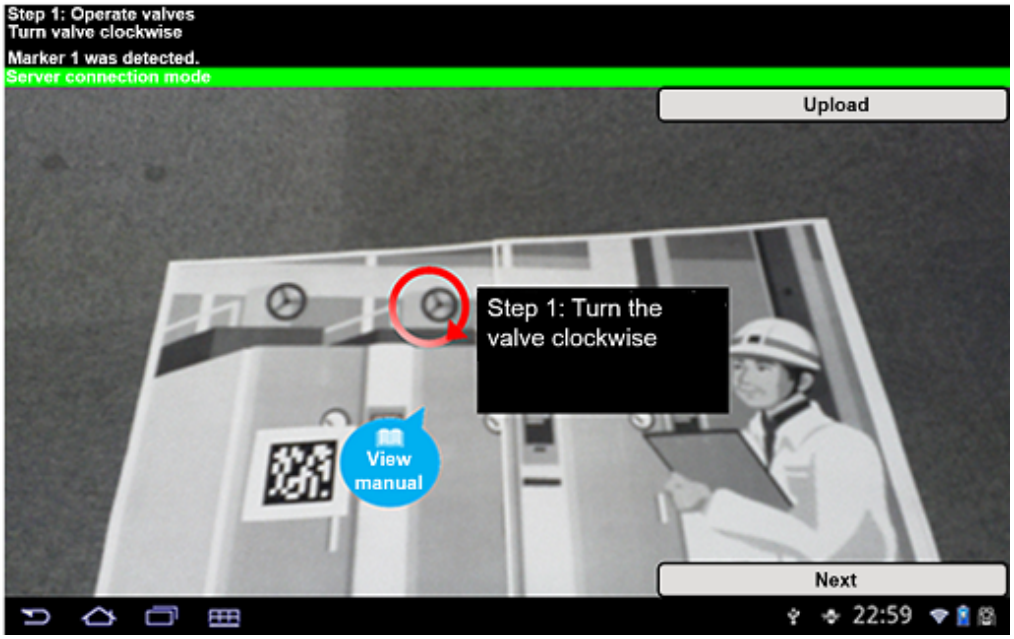
7.3.5 Authoring

Use server authoring from the data management console to create AR overlay definitions in the sample application (AR marker).

7.3.5.1 Operate valves scene

1. Click the **Inspection work** scenario.
2. On the scene list, click **Operate valves**.
3. Create the following AR overlay definitions:

Figure 7.2 Example overlay for operating valves



Add "Job sample_valve arrow"

Click **Add shape** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_valve arrow
	AR marker ID		1
Select shape	Shapes		
Position information	Projection		2D
	Translation	x	1.5
		y	2.5
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	0.2
		y	0.2
		z	0.2
Other	Use tap action		No

Add "Job sample_valve step"

Click **Add text** and enter the following content:

Item		Content to be entered
Basic information	AR overlay definition name	Job sample_Valve step
	AR marker ID	1
Text information	Text	Step 1: Turn valve clockwise

Item			Content to be entered
	Font size		20
	Font color		#[B00000] Transparency 255
	Background color		#[FFC0C0] Transparency 255
	Text wrapping		Enabled
	Area height		80
	Area width		160
Position information	Projection		2D
	Translation	x	4.0
		y	1.5
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	1.0
		y	1.0
		z	1.0
Other	Use tap action		Yes
	Script action		Apl.shiftScene(true)

Add "Job sample_manual"

Click **Add File** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_manual
	AR marker ID		1
Select file	URL		URL of MANUALPOP.png
Position information	Projection		2D
	Translation	x	1.0
		y	0.0
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	0.5
		y	0.5
		z	0.5
Other	Use tap action		Yes
	URL action		URL of MANUAL.pdf

7.3.5.2 Check interior scene

1. Click the **Inspection work** scenario.
2. On the scene list, click **Check interior**.

3. Create the following AR overlay definitions:

Figure 7.3 Example overlay for checking the interior

Add "Job sample_check int arrow"

Click **Add shape** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_check int arrow
	AR marker ID		1
Select shape	Shape		
Position information	Projection		2D
	Translation	x	0.0
		y	2.0
		z	-0.5
	Rotation	x	0.0
		y	0.0
		z	90.0
	Scale	x	0.2
		y	0.2
		z	0.2
Other	Use tap action		No

Add "Job sample_check interior step"

Click **Add text** and enter the following content:

Item		Content to be entered
Basic information	AR overlay definition name	Job sample_check int step
	AR marker ID	1

Item			Content to be entered
Text information	Text		Step 2: Open the door and check the interior
	Font size		20
	Font color		#[B00000] Transparency 255
	Background color		#[FFC0C0] Transparency 255
	Text wrapping		Enabled
	Area height		80
	Area width		160
Position information	Projection		2D
	Translation	x	4.0
		y	1.5
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	1.0
		y	1.0
		z	1.0
Other	Use tap action		Yes
	Script action		Apl.shiftScene(true)

Add "Job sample_check interior"

Click **Add File** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_check interior
	AR marker ID		1
Select file	URL		URL of MOVIEPOP.png
Position information	Projection		2D
	Translation	x	4.0
		y	-1.0
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	0.5
		y	0.5
		z	0.5
Other	Use tap action		Yes
	URL action		URL of MOVIE.mp4

7.3.5.3 Check value scene

1. Click the **Inspection work** scenario.

2. On the scene list, click **Check value**.
3. Create the following AR overlay definitions:

Figure 7.4 Example overlay for checking the value

Add "Job sample_check value step"

Click **Add text** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_check value step
	AR marker ID		1
Text information	Text		Step 3: Check the monitor value
	Font size		20
	Font color		#[B00000] Transparency 255
	Background color		#[FFC0C0] Transparency 255
	Text wrapping		Enabled
	Area height		80
	Area width		160
Position information	Projection		2D
	Translation	x	4.0
		y	1.5
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	1.0
		y	1.0
		z	1.0
Other	Use tap action		Yes

Item		Content to be entered
	Script action	Apl.shiftScene(true)

Add Job sample_check value

Click **Add File** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_check value
	AR marker ID		1
Select file	URL		URL of THUMBNAIL.jpg
Position information	Projection		2D
	Translation	x	2.0
		y	-1.0
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	0.5
		y	0.5
		z	0.5
Other	Use tap action		Yes
	URL action		URL of PICTURE.jpg

7.3.5.4 Enter temperature scene

1. Click the **Inspection work** scenario.
2. On the scene list, click **Enter temperature**.
3. Create the following AR overlay definitions:

Figure 7.5 Example overlay for entering the temperature

Add "Job sample_enter inspection val"

Click **Add File** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_enter inspect val
	AR marker ID		1
Select file	URL		URL of INPUT.png
Position information	Projection		2D
	Translation	x	1.5
		y	0.0
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	0.5
		y	0.5
		z	0.5
Other	Use tap action		Yes
	Script action		Apl.changeCommentpage()

Add "Job sample_enter inspection val step"

Click **Add text** and enter the following content:

Item			Content to be entered
Basic information	AR overlay definition name		Job sample_enter insp val step
	AR marker ID		1
Text information	Text		Step 4: Enter the temperature
	Font size		20
	Font color		#[B00000] Transparency 255
	Background color		#[FFC0C0] Transparency 255
	Text wrapping		Enabled
	Area height		80
	Area width		160
Position information	Projection		2D
	Translation	x	4.0
		y	1.5
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	1.0
		y	1.0
		z	1.0

Item		Content to be entered
Other	Use tap action	Yes
	Script action	Apl.shiftScene(true)

Add "Job sample_inspection value"

Click **Add text** and enter the following content:

Item			Content to be entered
Basic information	AR definition data name		Job sample_inspection value
	AR marker ID		1
Text information	Text		Not registered
	Font size		20
	Font color		#[000080] Transparency 255
	Background color		#[AFEEEE] Transparency 255
	Text wrapping		Enabled
	Area height		80
	Area width		160
Position information	Projection		2D
	Translation	x	4.0
		y	-1.5
		z	0.0
	Rotation	x	0.0
		y	0.0
		z	0.0
	Scale	x	1.0
		y	1.0
		z	1.0
Other	Use tap action		No

7.3.6 Create a user-defined table

Use the data management console to create the following user-defined table usr_sample.

Table 7.5 Attributes of the user-defined table usr_sample

Attribute name	Data type
ar_id	LONG
ar_name	STRING
ar_description	STRING
ar_registrationtime	LONG
ar_modificationtime	LONG
usr_name	STRING
usr_temperature	LONG

7.4 Running the sample application (AR marker)

Use the smart device to run the overlay application. The sample application (AR marker) enables you to trial online jobs and offline jobs.

If you click **AR Client Authoring** at startup, you can call the client authoring tool from the sample application (AR marker).

7.4.1 Printing the AR marker location sheet

Print the AR marker location sheet (background.pdf) and place the AR marker in the specified location.

7.4.2 Online jobs

Follow the procedure below to run online jobs:

1. Put the smart device online.
2. Start the sample native application that you installed on the smart device.
3. Tap **Overlay application**.
4. Enter the URL where you placed the overlay application to the **ServerURL** input form.

   Example

   - If using SSL encryption for communications on Model A:

     ```
     https://arServerName:httpsListenerPortForRestApi/webapp/index.html
     ```

   - If not using SSL encryption for communications on Model A:

     ```
     http://arServerName:httpListenerPortForRestApi/webapp/index.html
     ```

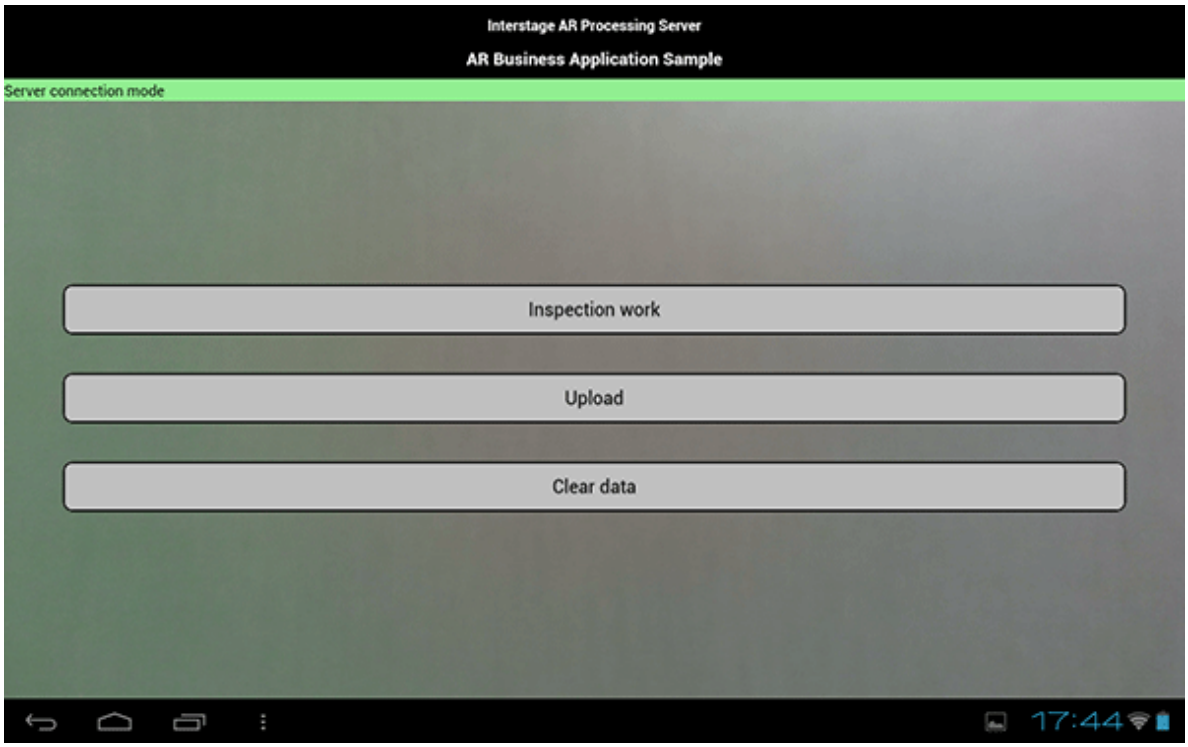
   - If using SSL encryption for communications on Model B/C/D:

     ```
     https://webServerName:webServerHttpsListenerPortForRestApi/webapp/index.html
     ```

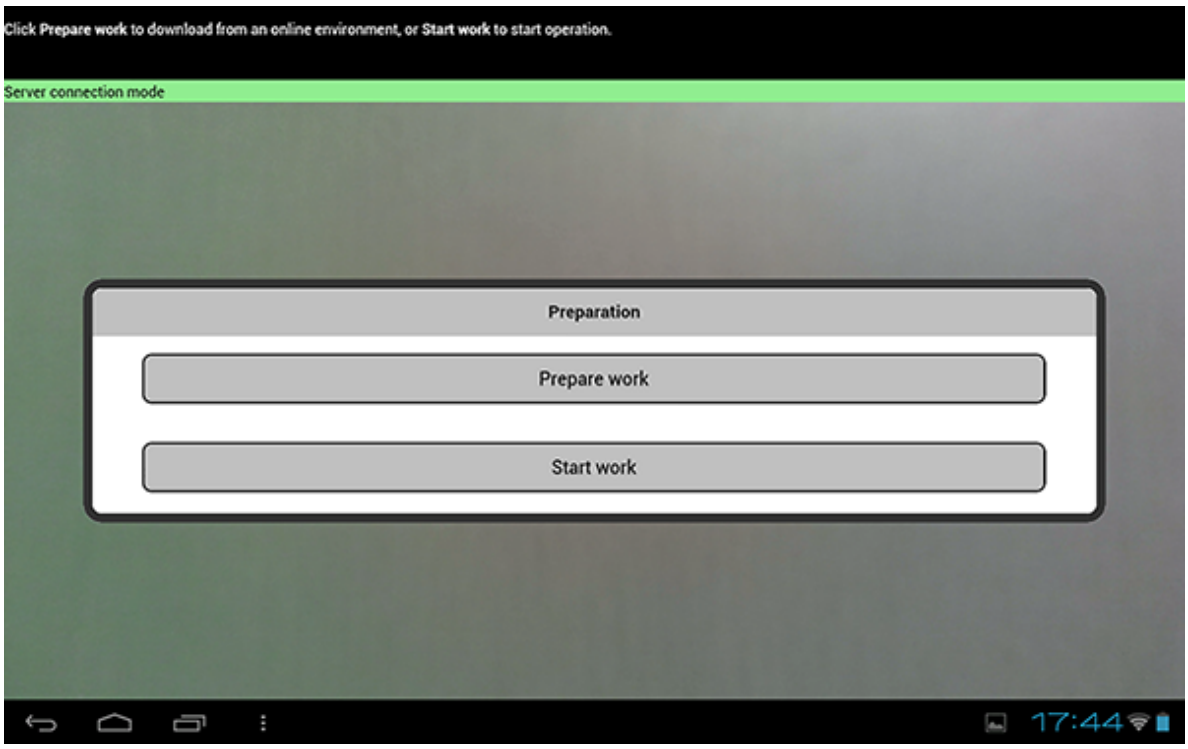
   - If not using SSL encryption for communications on Model B/C/D:

     ```
     http://webServerName :webServerHttpListenerPortForRestApi/webapp/index.html
     ```

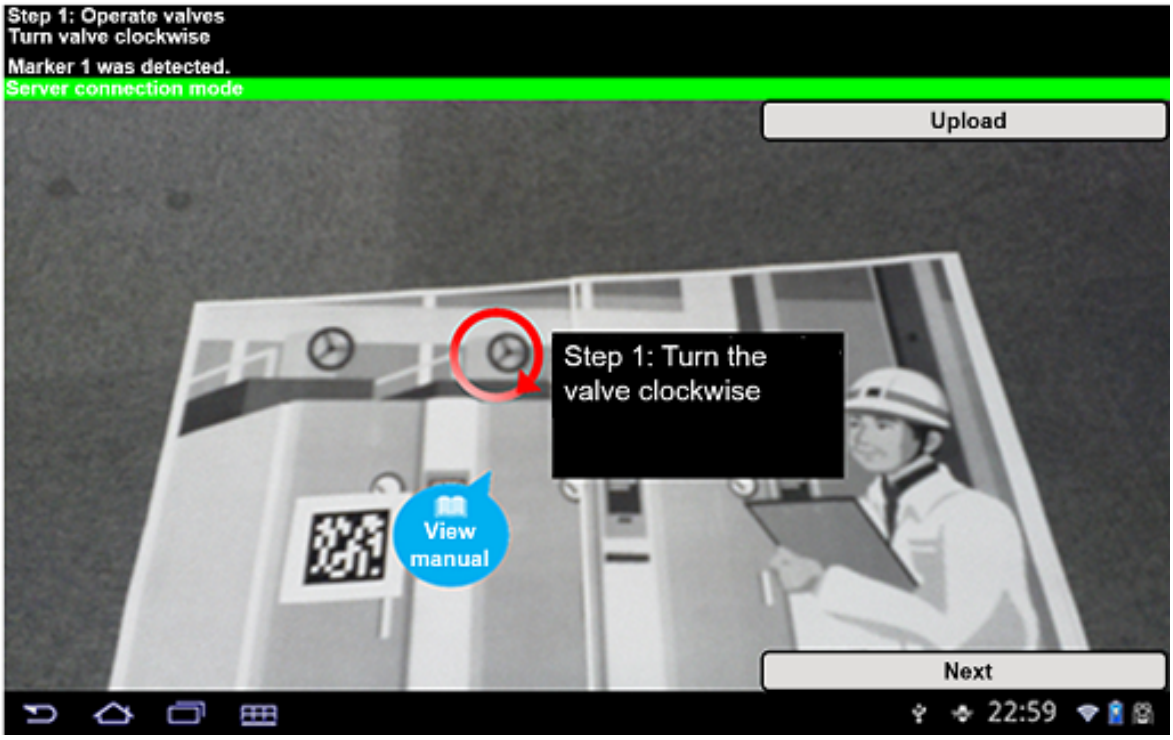
5. Tap **Start AR** and ensure that index.html is displayed.



6. In index.html, perform the following tasks:
   - Ensure that **Server Connection Mode** is displayed.
   - Click **Inspection work** to redirect to prework.html.
7. In prework.html, click **Start work**.



8. When work.html is displayed, perform the following tasks.



a. Scene 1

1. Hold the smart device up to the AR marker.
2. Ensure that **Step 1: Operate valves** is displayed at the top of the screen.
3. Tap **View manual** to display the manual.
4. Click **Next**.

b. Scene 2

1. Hold the smart device up to the AR marker.
2. Ensure that **Step 2: Check interior** is displayed at the top of the screen.
3. Tap **View video** to display a video.
4. Click **Next**.

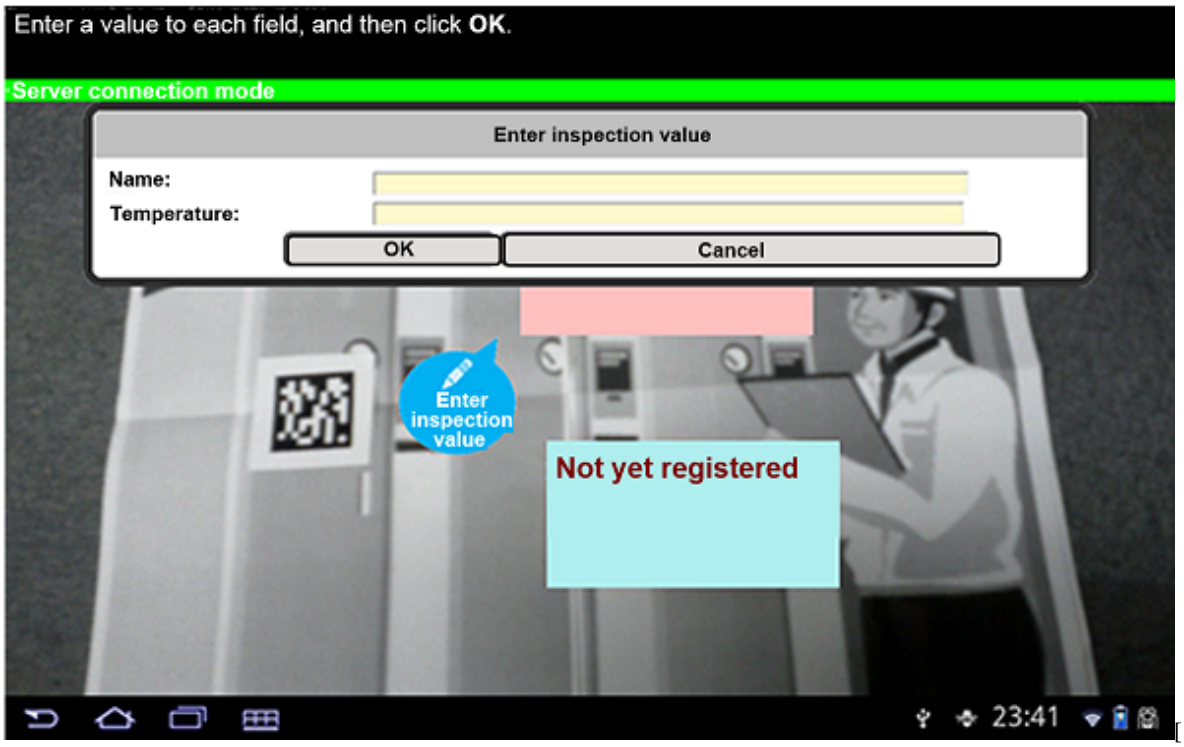
c. Scene 3

1. Hold the smart device up to the AR marker.
2. Ensure that **Step 3: Check value** is displayed at the top of the screen.
3. Tap **View image** to display an image.
4. Click **Next**.

d. Scene 4

1. Hold the smart device up to the AR marker.
2. Ensure that **Step 4: Enter temperature** is displayed at the top of the screen.
3. Tap **Enter inspection value** to redirect to comment.html.

9. In comment.html, enter the inspection value.



10. In work.html, ensure that the inspection result you just entered is overlaid.

7.4.3 Offline jobs

Follow the procedure below to run offline jobs:

1. Put the smart device online.
2. Start the sample application that you installed on the smart device.
3. Tap **Overlay application**.
4. Enter the URL where you placed the overlay application to the **ServerURL** input form.

   Example

   - If using SSL encryption for communications on Model A:

     ```
     https://arServerName:httpsListenerPortForRestApi/webapp/index.html
     ```

   - If not using SSL encryption for communications on Model A:

     ```
     http://arServerName:httpListenerPortForRestApi/webapp/index.html
     ```

   - If using SSL encryption for communications on Model B/C/D:

     ```
     https://webServerName:webServerHttpsListenerPortForRestApi/webapp/index.html
     ```

   - If not using SSL encryption for communications on Model B/C/D:

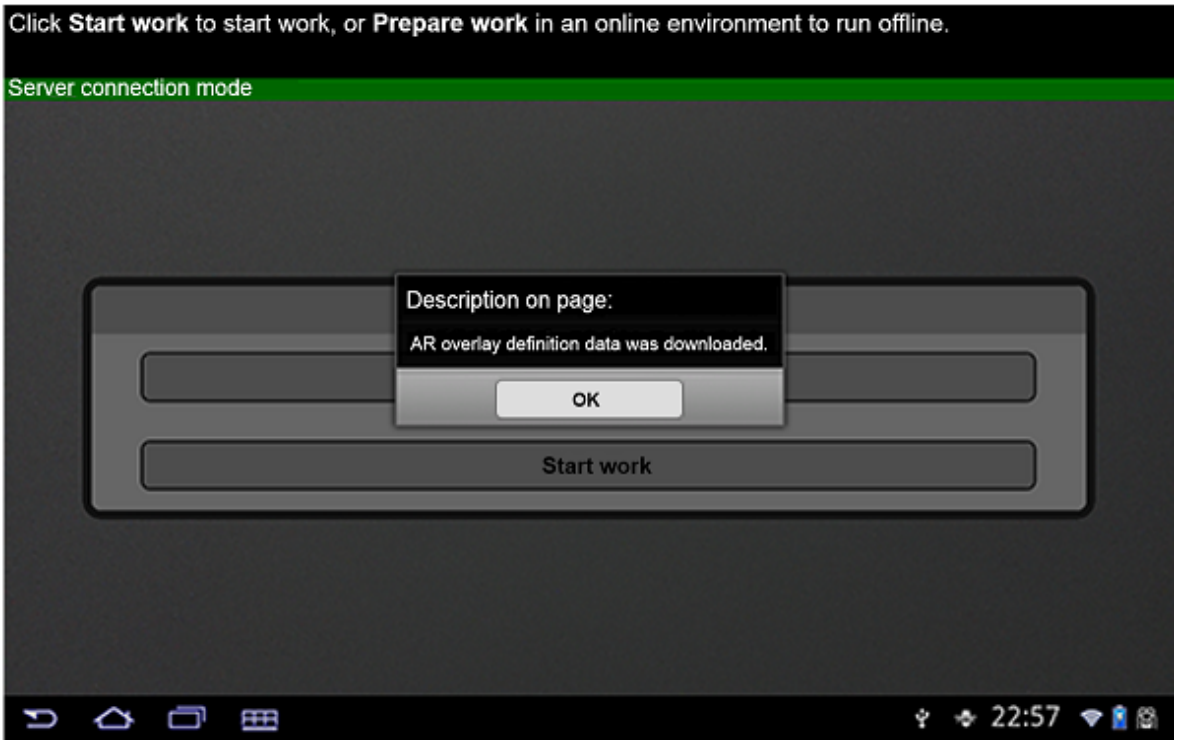
     ```
     http://webServerName:webServerHttpListenerPortForRestApi/webapp/index.html
     ```

5. Tap **Start AR** and ensure that index.html is displayed.
6. In index.html, perform the following tasks:

   ■ Ensure that **Server Connection Mode** is displayed.

- Click **Inspection work** to redirect to prework.html.

7. In prework.html, click **Prepare work**.
8. Ensure that a download message is displayed, and close the application.

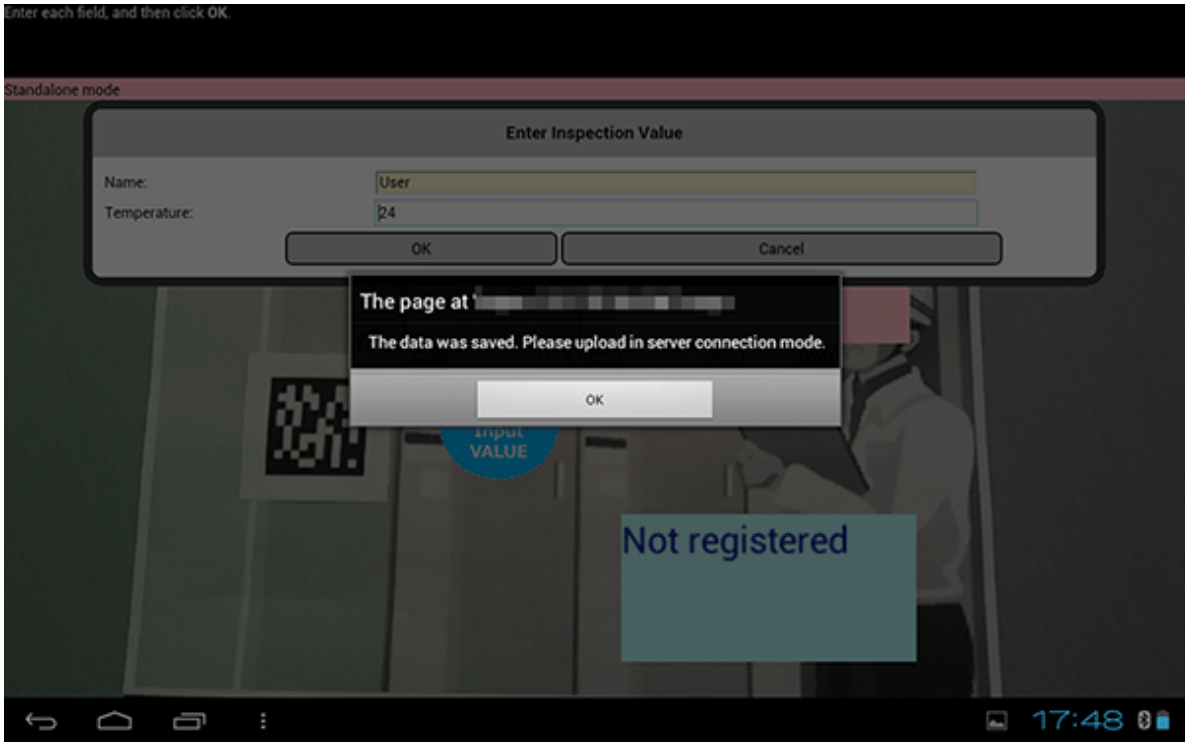


9. Set the smart device to flight mode to put it offline (refer to the relevant smart device manual for details).
10. Start the sample application (AR marker) again.
11. In index.html, perform the following tasks:
    - Ensure that **Stand-alone Mode** is displayed.
    - Click **Inspection work** to redirect to prework.html.
12. In work.html, perform the following tasks:
    a. Scene 1
        1. Hold the smart device up to the AR marker.
        2. Ensure that **Step 1: Operate valves** is displayed at the top of the screen.
        3. Tap **View manual** to display the manual.
        4. Click **Next**.
    b. Scene 2
        1. Hold the smart device up to the AR marker.
        2. Ensure that **Step 2: Check interior** is displayed at the top of the screen.
        3. Tap **View video** to display a video.
        4. Click **Next**.
    c. Scene 3
        1. Hold the smart device up to the AR marker.
        2. Ensure that **Step 3: Check value** is displayed at the top of the screen.
        3. Tap **View image** to display an image.
        4. Click **Next**.

d. Scene 4

1. Hold the smart device up to the AR marker.
2. Ensure that **Step 4: Enter temperature** is displayed at the top of the screen.
3. Tap **Enter inspection value** to redirect to comment.html.

13. In comment.html, enter the inspection value.
14. Click **OK** and ensure that a dialog box prompting you to upload the data in server connection mode is displayed.



Chapter 8 Sample application (Location Data)

This chapter explains the sample application that uses the Location Data and 3D model, focusing on the anticipated job content, the composition of the sample application, and how to run it.

8.1 Overview

8.1.1 Description

The sample application uses Location Data and a 3D model. The features are used as follows:

- Location Data
  Retrieves current positioning information and displays overlays based on the current site information.
- 3D model
  Performs display in an application according to the 3D model registration method.
- Screen capture
  Displays and captures the AR overlay content and camera image currently being displayed in the device.

8.1.2 Composition of the sample application (Location Data)

The sample application (Location Data) provided by Interstage AR Processing Server consists of the files listed below.

Point

The sample application provided by Interstage AR Processing Server is stored in the following location:

W *installDir*\samples\quickstart\3d_location

L /opt/FJSVar/samples/quickstart /

```
file:  Directory with files used for overlaying in the sample application
          bear.zip
takepicture:  Sample application directory
          apl_sample:  JavaScript directory for running the sample application
                    apl.js:  JavaScript file for running the sample application
          ar:  JavaScript library directory
                   ar.js:  JavaScript library file
          images:  Image directory
                  loading.png:  Image file
          WEB-INF:  Settings file directory for deploying the sample application
                web.xml:  Settings file
          index.html:  Application HTML file
          style.css:  Sample application css file
          work.appcache:  Application cache settings file
```

8.1.3 Features used by the sample application (Location Data)

The following table indicates which features provided by the JavaScript libraries are used by the sample application (Location Data).

Library	Feature	Used by the sample application (Location Data)
Camera control	Start/stop camera	N
	Register/remove marker detection	N
	Retrieve ID and rectangular coordinates of detected marker	N

Library	Feature	Used by the sample application (Location Data)
	Screen capture	Y
Log	Output log	Y
OS control	Report tap coordinate to native	N
	Display specified URL content	N
	Check operation mode	N
	Call settings screen	N
	Recognition mode settings	Y
AR rendering	Operate AR content: Retrieve	Y
	Operate AR content: Add	Y
	Operate AR content: Remove	N
	Operate AR content: Remove all	Y
Location Data control	Register/remove Location Data detection	Y
	Retrieve detected Location Data	Y
	Set radar for Location Data	Y
Data management control	Set AR processing server information	N
	Set use of offline storage	N
	Communicate with AR processing server: Retrieve	Y
	Communicate with AR processing server: Register	N
	Communicate with AR processing server: Remove	N
	Clear offline storage	N

8.2 Preparation

This section explains how to install and configure the server for running the sample application (Location Data).

8.2.1 Installing and configuring the server

Install and configure Interstage AR Processing Server - refer to the *Installation Guide* for details.

8.2.2 Deploying the sample application (Location Data)

Create a sample web application (Location Data) and deploy it so that it can be accessed from a smart device.

This example explains how to deploy the sample web application to the AR processing server. If deploying the sample web application to a model C or D configuration or other non-AR processing server environment, follow the procedure for deploying your application server.

1. Use the command prompt to redirect to the takepicture directory.
2. Use the jar command provided by JDK to create a 3d.war file. The directory structure when jar is deployed depends on the current directory, so redirect to the takepicture directory before running the command.
   Refer to the JDK manual provided by Oracle Corporation for details on jar.

Example

```
"C:\Program Files\Interstage\ISAR\components\STFISAR_HJDK7\jdk7\bin\jar.exe" -cvf  ..\3d.war *
```

3. Execute arsvadmin to deploy the 3d.war file.

Example

Deploying to the AR processing server

Execute arsvadmin to deploy the application:

```
"installDir\bin\arsvadmin" deploy ..\3d.war
```

Undeploying from the AR processing server

To cancel deployment after the sample is run, execute the following command:

```
"installDir\bin\arsvadmin" undeploy 3d
```

See

Refer to the *Operator's Guide* for details on deployment.

8.2.3 Installing the native application

Install the sample native application provided by Interstage AR Processing Server to a smart device.

The Android version of the native application is stored in the following location:

W *installDir*\mobileclient\android\ARClientDev.apk

L /opt/FJSVar/mobileclient/android/ARClientDev.apk

Example

Installing the Android version of the native application via a USB connection

The example below installs the application on the smart device by connecting to a PC via USB (the USB connection method varies between smart devices - refer to the smart device manual for details).

1. Copy the sample native application to a PC that can be connected via USB to a smart device.
2. Select **Applications from unknown source** on the smart device.
   - On the **Settings** screen, tap **Security**, and then select **Unknown sources**.
3. Connect the smart device to the PC using a USB cable. If the PC does not recognize the driver, then install it.
4. On the PC, select the smart device using Windows Explorer - if the PC does not recognize the smart device, refer to the smart device manual.
5. Copy ARClientDev.apk to the desired location in the smart device.
6. Close the Windows Explorer.
7. On the PC taskbar, select **Safely remove hardware and eject media** and release the mounted device.
8. Start the application with the file management feature and select ARClientDev.apk. If there is no application with a file management feature, then install one.
9. The installation screen will be displayed. Start the installation.
10. Upon completion, on the **Settings** screen, **Security** menu, clear **Unknown sources**.

8.2.4 Configuring the native application

Configure the native application. Refer to the *Installation Guide* for details.

Point

In the AR server account settings explained in the *Installation Guide*, set the AR processing server account used in the sample application.

8.3 Data management

Use the data management console to register data in the system. Follow the procedure below to register data:

1. Register a scenario
   Register data for the scenario used in the sample application (Location Data).
2. Register scenes
   Register the scenes used in the sample application (Location Data).
3. Register files
   Register the image files to be used in the sample application (Location Data).
4. Authoring
   Register AR overlay definitions to be used in the sample.

See

Refer to the *Operator's Guide* for details on how to use the data management console.

Note

- If you enter different values for the data to be used by the sample application (Location Data), the application may not operate properly. Enter the data exactly as given in this chapter.
- You can change the scenario ID to be used by changing the value of Apl.useScenarioId in apl.js.
- Do not change the scene IDs.

8.3.1 Registering a scenario

Use the data management console to register the following scenario data.

Table 8.1 Content to be entered for the scenario

Item	Content to be entered
Scenario ID	901
Scenario name	Location Data 3D sample
Remarks	-

8.3.2 Registering scenes

Use the data management console to register the following scene for the **Location Data 3D sample** scenario.

Table 8.2 Scene registration

Item	Content to be entered
Scene ID	9011
Scene name	Screen capture

Item	Content to be entered
Remarks	-

8.3.3 Authoring

Create the following content in the registered **Screen capture** scene.

Add 3D model

In **AR overlay definition list**, click **Add 3D Model** and enter the following content:

Item			Content to be entered
Basic information	Detection method		Location Data
	AR overlay definition name		Bear
Select 3D file	Selection Method		Upload from local system
	Target		bear.zip
	MIME type		application/zip
Position Information	Projection		3D
	Positioning information	Latitude	0
		Longitude	0
		Altitude	0
	Rotation	x (tilt)	0.0
		y (azimuth)	0.0
		z (rotational angle)	0.0
	Scale	x	0.5
		y	0.5
		z	0.5
Other	Use tap action		No

Note

The value entered for Location Data in the sample application is "0". Before actually using Location Data for authoring, retrieve the Location Data of the location over which the AR overlay should be displayed.

8.4 Running the sample application

8.4.1 Executing the sample application (Location Data)

Follow the procedure below to run the sample application (Location Data).

1. Put the smart device online and ensure that the Location Data has been retrieved.
2. Start the sample application that you installed on the smart device.
3. Tap **Overlay application**.
4. Enter the URL where you placed the AR overlay application to the **Server URL** input form.
5. Tap **Start AR** and ensure that index.html is displayed.

8.4.2 Results

The index screen displays the following information:

Figure 8.1 Example of running the sample

- Status
  - Status
    Displays the status of Location data retrieval.
  - Location data
    Displays latitude, longitude, and altitude in decimal (only displayed if this information can be retrieved).
  - Camera orientation
    Displays the angle of tilt, azimuth, and rotational angle in degrees (only displayed if this information can be retrieved).
- Radar
  Displays a radar showing the location of the user, the location of content, and the north direction.
- Get 3D model
  Retrieves the specified 3D model and displays it in the foreground. If you run the sample while a 3D model is already displayed, the displayed 3D model will be deleted and replaced with the latest 3D model.

Note

  - Depending on the accuracy of the Location Data retrieved, the overlaid content may extend beyond the display range.
  - The accuracy of the Location Data retrieved depends on the smart device used. Refer to the *Operator's Guide* for details.

- Take picture
  Takes a snapshot and saves the image currently captured by the camera, including any 3D model being displayed.

Chapter 9 Sample application (barcode)

This chapter explains the sample application that uses barcodes, focusing on the composition of the sample application, and how to run it.

9.1 Overview

9.1.1 Description

The sample application uses barcodes. This section explains how to recognize each format and how to add a listener.

9.1.2 Composition of the sample application (barcode)

The application consists of the files listed below.

The sample application provided by Interstage AR Processing Server is stored in the following location:

Example

W *installDir*\samples\quickstart\

L /opt/FJSVar/samples/quickstart/

```
barcode: Sample application directory

        apl_sample: JavaScript directory for running the sample application

                       apl.js: JavaScript file for running the sample application

        ar:  JavaScript library directory

            ar.js:  JavaScript library file

        images: Image directory

               checkbox-checked.png: Image file

               checkbox-uncheck.png: Image file

        WEB-INF: Settings file directory for deploying the sample application

                web.xml: Settings file

        index.html: Application HTML file

        style.css: Sample application CSS file

        work.appcache: Application cache settings file
```

9.1.3 Features used by the sample application (barcode)

Library	Feature	Used by the sample application (barcode)
Camera control	Start/stop camera	Y
	Register/remove marker detection	N

Library	Feature	Used by the sample application (barcode)
	Retrieve ID and rectangular coordinates of detected marker	N
	Screen capture	N
Log	Output log	Y
OS control	Report tap coordinate to native	Y
	Display specified URL content	N
	Check operation mode	Y
	Call settings screen	N
	Recognition mode settings	Y
AR rendering	Operate AR content: Retrieve	N
	Operate AR content: Add	Y
	Operate AR content: Remove	N
	Operate AR content: Remove all	N
Barcode control	Register/remove barcode detection	Y
	Retrieve ID and rectangular coordinates of detected barcode	Y
Data management control	Set AR processing server information	N
	Set use of offline storage	N
	Communicate with AR processing server: Retrieve	Y
	Communicate with AR processing server: Register	N
	Communicate with AR processing server: Remove	N
	Clear offline storage	N

9.2 Preparation

9.2.1 Installing and configuring the server

Install and configure Interstage AR Processing Server - refer to the *Installation Guide* for details.

9.2.2 Deploying the sample application (barcode)

Create a sample web application (barcode) and deploy it so that it can be accessed from a smart device.

This example explains how to deploy the sample web application to the AR processing server. If deploying the sample web application to a model C or D configuration or other non-AR processing server environment, follow the procedure for deploying your application server.

1. Use the command prompt to redirect to the barcode directory.
2. Use the jar command provided by JDK to create a barcode.war file. The directory structure when jar is deployed depends on the current directory, so redirect to the barcode directory before running the command.
   Refer to the JDK manual provided by Oracle Corporation for details on jar.

Example

```
"C:\Program Files\Interstage\ISAR\components\STFISAR_HJDK7\jdk7\bin\jar.exe" -cvf  ..
\barcode.war *
```

3. Execute arsvadmin to deploy the barcode.war file.

Example

Deploying to the AR processing server

Execute arsvadmin to deploy the application:

```
"installDir\bin\arsvadmin" deploy barcode.war
```

Undeploying from the AR processing server

To cancel deployment after the sample is run, execute the following command:

```
"installDir\bin\arsvadmin" undeploy barcode
```

See

Refer to the *Operator's Guide* for details on deployment.

9.2.3 Installing the native application

Install the sample native application provided by Interstage AR Processing Server to a smart device.

The Android version of the native application is stored in the following location:

W *installDir*\mobileclient\android\ARClientDev.apk

L /opt/FJSVar/mobileclient/android/ARClientDev.apk

Example

Installing the Android version of the native application via a USB connection

The example below installs the application on the smart device by connecting to a PC via USB (the USB connection method varies between smart devices - refer to the smart device manual for details).

1. Copy the sample native application to a PC that can be connected via USB to a smart device.
2. Select **Applications from unknown source** on the smart device.
   - On the **Settings** screen, tap **Security**, and then select **Unknown sources**.
3. Connect the smart device to the PC using a USB cable. If the PC does not recognize the driver, then install it.
4. On the PC, select the smart device using Windows Explorer - if the PC does not recognize the smart device, refer to the smart device manual.
5. Copy ARClientDev.apk to the desired location in the smart device.
6. Close the Windows Explorer.
7. On the PC taskbar, select **Safely remove hardware and eject media** and release the mounted device.
8. Start the application with the file management feature and select ARClientDev.apk. If there is no application with a file management feature, then install one.
9. The installation screen will be displayed. Start the installation.
10. Upon completion, on the **Settings** screen, **Security** menu, clear **Unknown sources**.

9.2.4 Configuring the native application

Configure the native application. Refer to the *Installation Guide* for details.

Point

In the AR server account settings explained in the *Installation Guide*, set the AR processing server account used in the sample application.

9.3 Data management

The sample application superimposes the overlay content associated with the barcode for scenario ID 902, scene ID 1 in the application. Before running the sample application, the content must be registered in the system using the data management console (refer to the *Operator's Guide* for details).

9.4 Running the sample application

9.4.1 Executing the sample application (barcode)

Follow the procedure below to run the sample application (barcode):

1. Put the smart device online.
2. Start the sample application that you installed on the smart device.
3. Tap **Overlay application**.
4. Enter the URL where you placed the AR overlay application to the **Server URL** input form.
5. Tap **Start AR** and ensure that index.html is displayed.

9.4.2 Results

The index screen displays the following information:

Figure 9.1 Example of running the sample

- Status
  - Displays the barcode ID (only displayed if a listener has been added).
- Detect barcode type
  - Enables detection of the formats selected from the check boxes.
- Add listener
  - Adds a listener.
- Barcode List
  - Displays the list of recognized barcodes.
- Listener list
  - Displays the list of active listeners. Registered listeners can be removed from this list.

Chapter 10 Sample application (beacon)

This chapter explains the sample application that uses beacons, focusing on the composition of the sample application, and how to run it.

10.1 Overview

10.1.1 Description

The sample application uses beacons. This section explains how to add a listener and how to check the beacon recognition status.

10.1.2 Composition of the sample application (beacon)

The application consists of the files listed below.

The sample application provided by Interstage AR Processing Server is stored in the following location:

W *installDir*\samples\quickstart\

L /opt/FJSVar/samples/quickstart/

```
beacon: Sample application directory

       apl_sample: JavaScript directory for running the sample application

                     apl.js: JavaScript file for running the sample application

       ar:  JavaScript library directory

              ar.js:  JavaScript library file

       WEB-INF: Settings file directory for deploying the sample application

                  web.xml: Settings file

       index.html: Application HTML file

       style.css: Sample application CSS file

       work.appcache: Application cache settings file
```

10.1.3 Features used by the sample application (beacon)

Library	Feature	Used by the sample application (beacon)
Camera control	Start/stop camera	Y
	Register/remove marker detection	N
	Retrieve ID and rectangular coordinates of detected marker	N
	Screen capture	N
Log	Output log	Y
OS control	Report tap coordinate to native	Y
	Display specified URL content	N
	Check operation mode	Y
	Call settings screen	N

Library	Feature	Used by the sample application (beacon)
	Recognition mode settings	Y
AR rendering	Operate AR content: Retrieve	N
	Operate AR content: Add	Y
	Operate AR content: Remove	N
	Operate AR content: Remove all	N
Beacon control	Register/remove beacon detection	Y
	Retrieve UUID and rectangular coordinate of detected beacon	Y
Data management control	Set AR processing server information	N
	Set use of offline storage	N
	Communicate with AR processing server: Retrieve	Y
	Communicate with AR processing server: Register	N
	Communicate with AR processing server: Remove	N
	Clear offline storage	N

10.2 Preparation

10.2.1 Installing and configuring the server

Install and configure Interstage AR Processing Server - refer to the *Installation Guide* for details.

10.2.2 Deploying the sample application (beacon)

Create a sample web application (beacon) and deploy it so that it can be accessed from a smart device.

This example explains how to deploy the sample web application to the AR processing server. If deploying the sample web application to a model C or D configuration or other non-AR processing server environment, follow the procedure for deploying your application server.

1. Use the command prompt to redirect to the beacon directory.
2. Use the jar command provided by JDK to create a beacon.war file. The directory structure when jar is deployed depends on the current directory, so redirect to the beacon directory before running the command.
   Refer to the JDK manual provided by Oracle Corporation for details on jar.

   Example

   ```
   "C:\Program Files\Interstage\ISAR\components\STFISAR_HJDK7\jdk7\bin\jar.exe" -cvf  ..\beacon.war
   *
   ```
3. Execute arsvadmin to deploy the beacon.war file.

Example

Deploying to the AR processing server

Execute arsvadmin to deploy the application:

```
"installDir\bin\arsvadmin" deploy beacon.war
```

Undeploying from the AR processing server

To cancel deployment after the sample is run, execute the following command:

```
"installDir\bin\arsvadmin" undeploy beacon
```

See

Refer to the *Operator's Guide* for details on deployment.

10.2.3 Installing the native application

Install the sample native application provided by Interstage AR Processing Server to a smart device.

The Android version of the native application is stored in the following location:

W *installDir*\mobileclient\android\ARClientDev.apk

L /opt/FJSVar/mobileclient/android/ARClientDev.apk

Example

Installing the Android version of the native application via a USB connection

The example below installs the application on the smart device by connecting to a PC via USB (the USB connection method varies between smart devices - refer to the smart device manual for details).

1. Copy the sample native application to a PC that can be connected via USB to a smart device.
2. Select **Applications from unknown source** on the smart device.
   - On the **Settings** screen, tap **Security**, and then select **Unknown sources**.
3. Connect the smart device to the PC using a USB cable. If the PC does not recognize the driver, then install it.
4. On the PC, select the smart device using Windows Explorer - if the PC does not recognize the smart device, refer to the smart device manual.
5. Copy ARClientDev.apk to the desired location in the smart device.
6. Close the Windows Explorer.
7. On the PC taskbar, select **Safely remove hardware and eject media** and release the mounted device.
8. Start the application with the file management feature and select ARClientDev.apk. If there is no application with a file management feature, then install one.
9. The installation screen will be displayed. Start the installation.
10. Upon completion, on the **Settings** screen, **Security** menu, clear **Unknown sources**.

10.2.4 Configuring the native application

Configure the native application. Refer to the *Installation Guide* for details.

Point

In the AR server account settings explained in the *Installation Guide*, set the AR processing server account used in the sample application.

10.3 Data management

The sample application superimposes the overlay content associated with the beacon for scenario ID 903, scene ID 1 in the application. Before running the sample application, the content must be registered in the system using the data management console (refer to the *Operator's Guide* for details).

10.4 Running the sample application

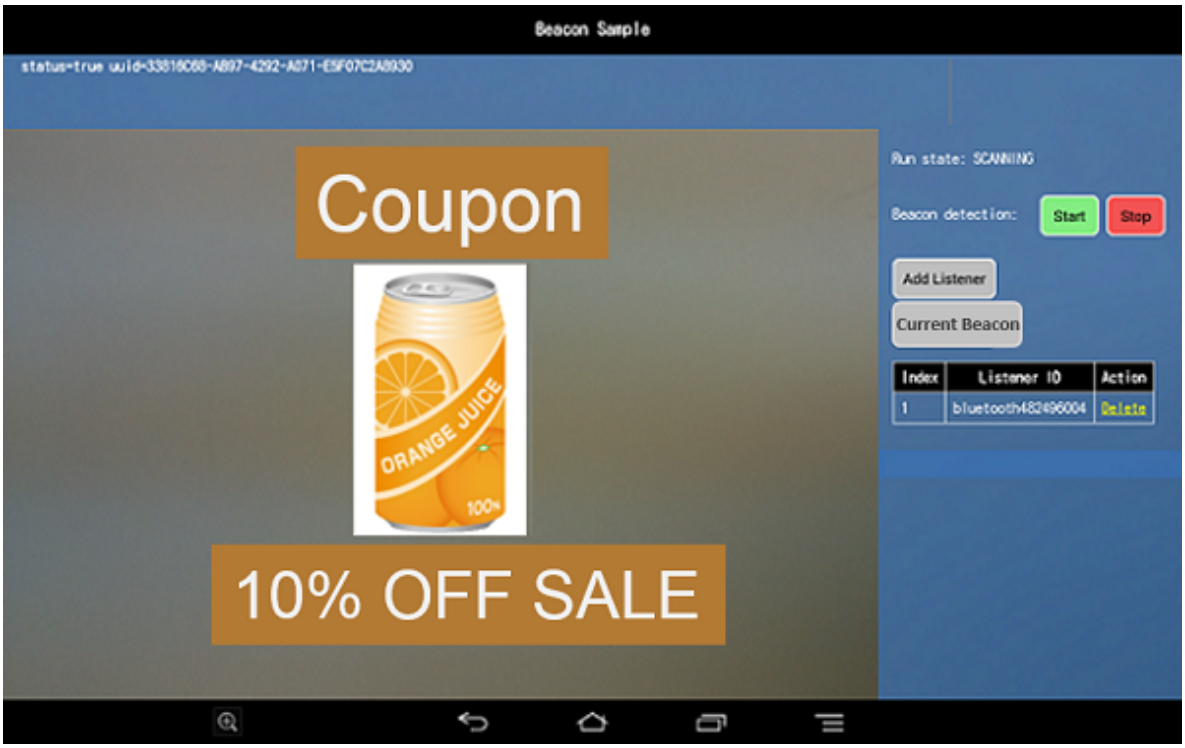
10.4.1 Executing the sample application (beacon)

1. Put the smart device online.
2. Start the sample native application that you installed on the smart device.
3. Select **Overlay application**.
4. Enter the URL where you placed the overlay application to the **Server URL** input form.
5. Tap **Start AR** and ensure that index.html is displayed.

10.4.2 Results

The index screen displays the following information:

Figure 10.1 Example of running the sample



- Status
  - Displays the UUID (only displayed if it has been added to a listener).

- Operation status
  - Displays the operation status of the beacon detection feature of the device.
- Beacon detection
  - Starts/stops beacon detection.
- Add listener
  - Adds a listener.
- Beacon list
  - Displays the list of the recognized beacons
- Listener list
  - Displays the list of active listeners. Registered listeners can be removed from this list.