IMiS®/wScan Manual

Version 1.3.1802

IMAGING SYSTEMS

Imaging Systems Inc.
Brnciceva 41 G
Ljubljana
Slovenia

TABLE OF CONTENTS

TABLE OF IMAGES

Table of images appearing in the manual

1 INTRODUCTION

1.1 About manual

This IMiS®/wScan manual describes the functionalities of and working with the IMiS®/wScan application.

1.2 Target audience

It is intended for administrators and application developers with prior technical knowledge who need information about the installation and configuration of the IMiS®/wScan application, its IMiS®/CaptureService core, and integration with third-party applications.
Below is the technical documentation with a detailed description of IMiS®/Capture Service, provided for application developers.

2 GENERAL

IMiS®/wScan is up to date with all contemporary technological, functional and design standards in the field of software for the capture of physical documents. Its design enables fully functional use in a multi-level architecture, with a web browser as the integration point. Owing to its modular and level design it is highly customizable and usable in various implementation scenarios, either with or without user interaction.

Despite the fact that the capture of physical content can hardly be done without using physical computer components (optical scanner), the use of which has practically been prevented by web browser producer, IMiS®/wScan nevertheless enables the digitization of physical documents by using inherently safe technologies in a purely online solution without the use of plug-ins or similar components.

It enables users to capture contents and digitize them in pure online solutions. It has been made in compliance with the ECMAScript 2016 specification. Despite the rather new JavaScript language specification, browser support is guaranteed sufficiently.

The web service meets the following criteria, conceptually:

- It has been designed with pure JavaScript technology without any additional requirements for e.g. plug-ins or access to the “native” protocols NPAPI, COM, etc. Therefore, it is not based on the technologies that browser makers consider unsuitable (dangerous).
- This simple, intuitive and customizable user interface enables application developers’ full flexibility when integrating it into any web applications.
- Integration at a sufficiently low level enables application developers’ customization even in the event of technological conflicts with an internally used framework should the application developer be unable to use e.g. constructs of the View level of the application.

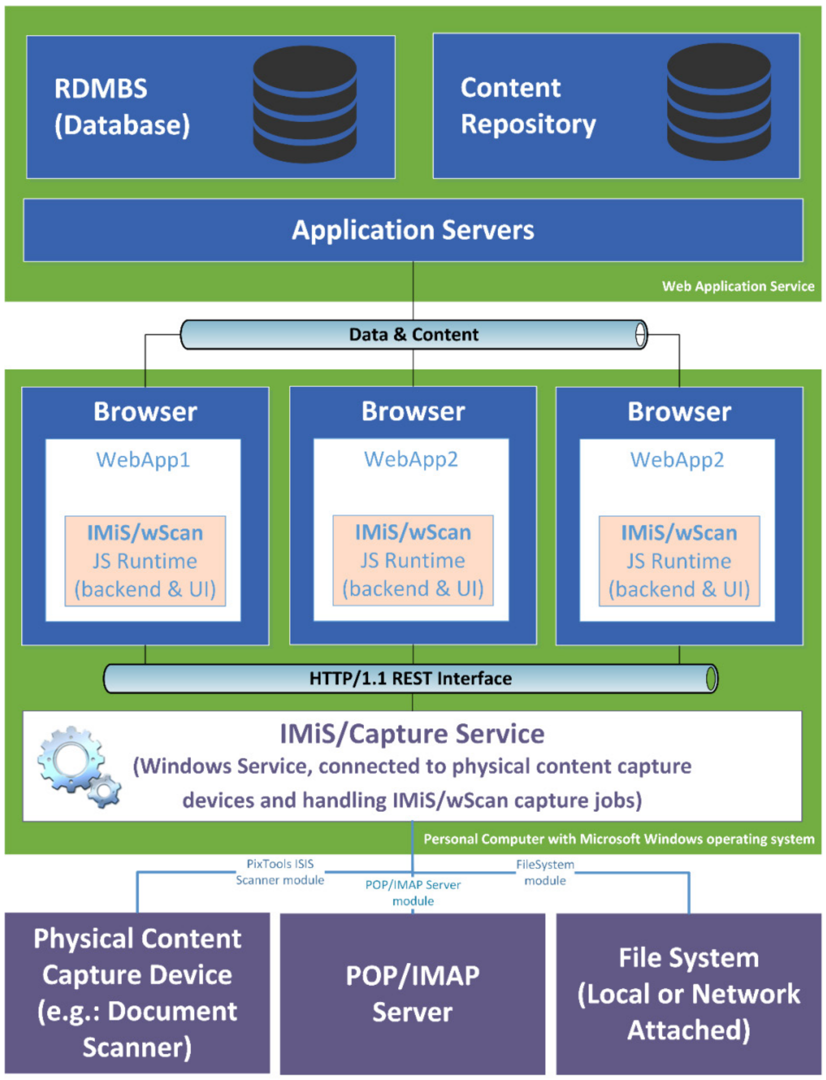

RDMBS (Database)
Content Repository
Application Servers
Web Application Service
Data & Content
Browser
WebApp1
IMiS/wScan JS Runtime (backend & UI)
Browser
WebApp2
IMiS/wScan JS Runtime (backend & UI)
Browser
WebApp2
IMiS/wScan JS Runtime (backend & UI)
HTTP/1.1 REST Interface
IMiS/Capture Service
(Windows Service, connected to physical content capture devices and handling IMiS/wScan capture jobs)
Personal Computer with Microsoft Windows operating system
PixTools ISIS Scanner module
POP/IMAP Server module
FileSystem module
Physical Content Capture Device (e.g.: Document Scanner)
POP/IMAP Server
File System (Local or Network Attached)

Image 1: Scheme of integrating the product into a multi-level web application

2.1 Architecture

2.1.1 Modular design

IMiS®/Capture Service has a module chain design in which different monolithic building blocks (modules) can be arranged in sequence, depending on the needs of the job. Each module provides a certain functionality (e.g. module for communicating with an optical scanner, module for barcode recognition, module for separating documents, module for merging documents, module for document conversion, module for saving to the archive system, etc.). The modular design enables capture from a different and heterogeneous selection of sources.

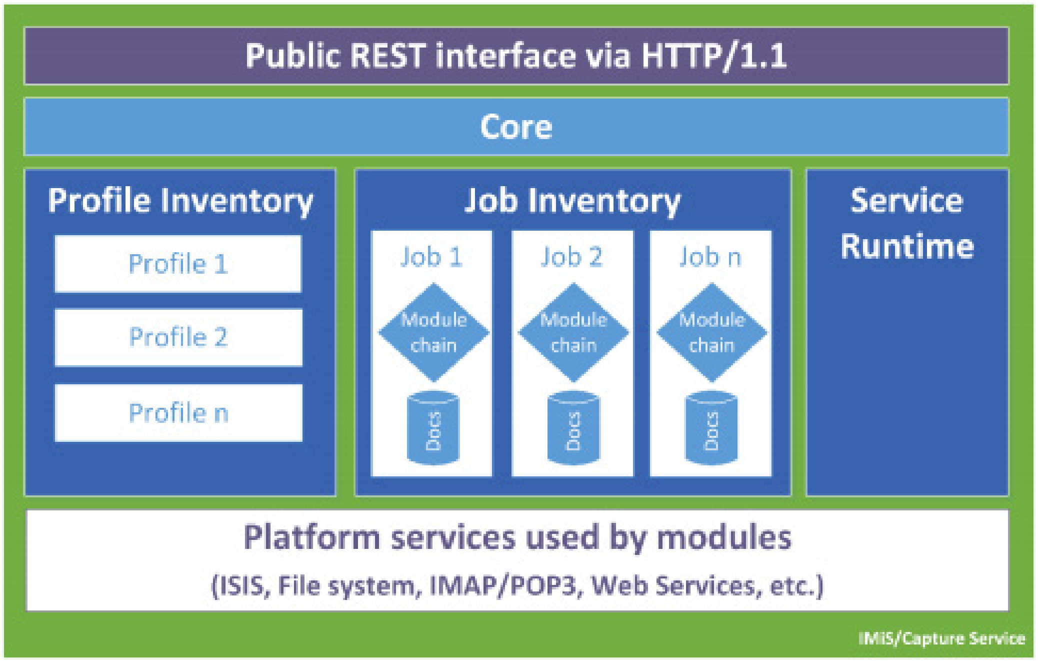


Image 2: Scheme of the IMiS®/Capture Service core service

Note: Further development anticipates a gradual introduction of modules for the capture of documents from inboxes via IMAP/POP3 technology, a file system, external sources through various web services, etc. The main thing is that this will be a single point of capturing and processing (separating logical documents by barcodes, etc.) for all input documents regardless of the channel used for the capture.

2.1.2 Multi-level architecture

IMiS®/Capture Service is a Windows-compatible backend service for the document capture and control of connected ISIS-compatible optical scanners. These functions make use of the Captiva PixTools technology (http://documentum.opentext.com/captiva-oem/software/pixtools-toolkit/) in its Microsoft .NET implementation. In order to provide the widest possible range of options for integration into various technologies, this service has been designed without a user interface. Its functionality is fully utilized via its REST interface. The latter is accessible via .NET web server, build-into the service and is based on Hypertext Transfer Protocol -- HTTP/1.1 technology (https://www.w3.org/Protocols/rfc2616/rfc2616.html).

It functionally provides the service of lifecycle management of:

- Capture settings (the so-called Profile Lifecycle), which can be used for capture (a profile is the saved settings of a document capture job).
- Capture jobs (the so-called Job Lifecycle).
- An individual captured document (the so-called Document Lifecycle).

The program interface of the IMiS®/wScan product is simple, intuitive and highly customizable. It enables application developers to fully adapt the functionalities to arbitrary web applications. It has been designed as a multi-level interface with pure JavaScript technology that does not require external JavaScript frameworks (e.g. plug-ins or access to the “native” protocols NPAPI, COM, etc.).
Through integration developers will be able to descend to a sufficiently low level that will enable the necessary customizations. This will come especially in handy when the user will be unable to use e.g. constructs of the View (UI) level of the application due to technological conflicts with an internally used framework.

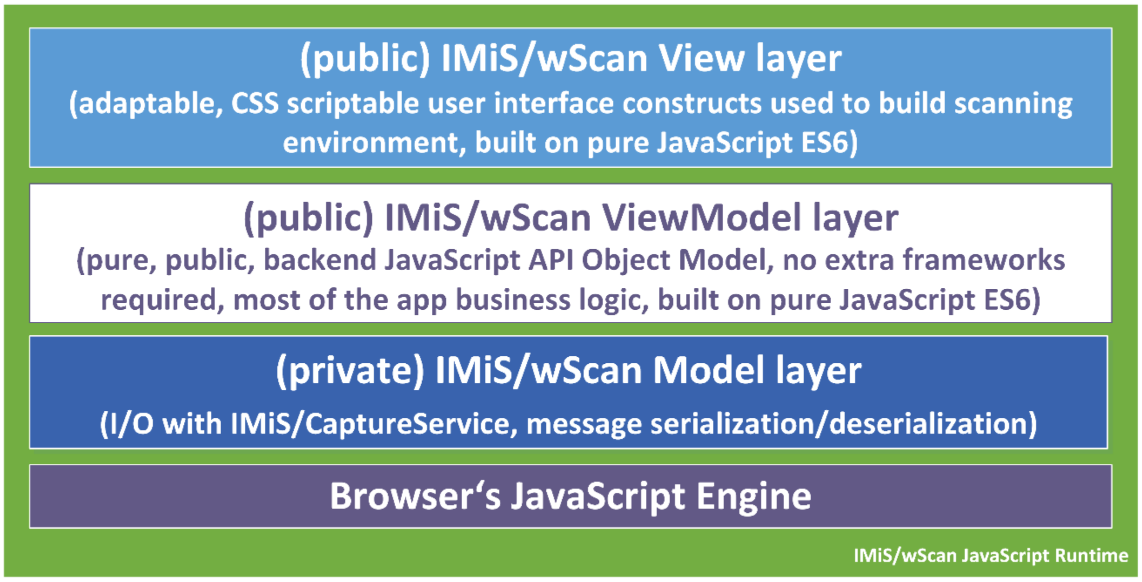

(public) IMiS/wScan View layer
(adaptable, CSS scriptable user interface constructs used to build scanning environment, built on pure JavaScript ES6)
(public) IMiS/wScan ViewModel layer
(pure, public, backend JavaScript API Object Model, no extra frameworks required, most of the app business logic, built on pure JavaScript ES6)
(private) IMiS/wScan Model layer
(I/O with IMiS/CaptureService, message serialization/deserialization)
Browser's JavaScript Engine
IMiS/wScan JavaScript Runtime

Image 3: Scheme of JavaScript program levels of IMiS®/wScan application

The **Browser's JavaScript Engine** is the first and most basic level for communication with physical system components.

The **Model level** of the application is a private part of the application and is not intended for points of integration with applications. It connects it with IMiS®/Capture Service via an HTTP link (usually local, but can be remote). It provides support to the ViewModel level. It establishes externally an internal (private) object model, in which all service messages have been deserialized and made available to higher levels in the form of JavaScript objects. It also manages the asynchronous triggering of events originating from events on the service level, which higher levels would not be able to detect and respond to. Through this level all data and commands for operations are exchanged with the service part. Even though this level has been declared private, it has an open source code that can be viewed by external developers, especially during the development of the application, when they can track errors to this level as well.

The **ViewModel** level of the application is the heart of the application. To higher levels (and optionally to application developers) it exposes a rich and intuitive JavaScript object model with all of the business logic that ensures the consistency of JavaScript objects and the status of the service it is communicating with.

It is an entirely backend level without any user interface constructs. Its objects enable applications to manage IMiS®/Capture Service, manage the lifecycle of captured documents, etc. Its events model enables the coordination and synchronization of events that originate from the IMiS®/wScan application or from IMiS®/Capture Service.
For capture scenarios in which a user interface is not wanted or needed, the level has been designed to be used without a user interface in a way that does not limit its range of functionalities. These are the more exceptional events enabled by the architecture.

The **View level** of the application, as the last of the set, comprises a selection of user interface constructs that round off the application for the capture and digitization of physical documents. It is connected with the ViewModel level. It enables application developers easy and customizable integration of constructs into the application without possessing exact knowledge of the events and objects that enable the construct its function. The basic appearance of constructs can be customized via their object model (properties) or via CSS styles, with which the developer can customize the details of constructs to the requests and requirements of applications.
Similarly, to the other levels, this level has been built without requiring any additional JavaScript frameworks (e.g. AngularJS, etc.), which is why its integration does not cause conflicts with applications.
Constructs of the View model are conceptually and functionally independent of one another; however, for consistency purposes, the displayed information is interconnected via a network of events, which are forwarded from/to the ViewModel level. These make sure they are consistent as regards their content and status (example of displaying scanning progress).
The events of creating new pages in a document originate from the document capture service. The creation of each page has to be propagated as an event to the visual controls. The latter are connected to the ViewModel level via the events model; the ViewModel level is connected to the Model level and the latter is asynchronously connected to the Service level using WebSocket technology (otherwise it would have to execute service requests in intervals to inquire about status). Such an event is propagated from the Service level to the client's Model level; it in turn forwards it to the ViewModel level, which then refreshes all of the visual constructs that are subscribed to such an event. These constructs are programmed to call such a new page from the service and add it to the displayed list of pages.

2.2 Security

The basic installation does not foresee encryption of the web traffic of the REST interface, because traffic takes place locally via the local network interface "localhost", which means that protection is generally not required. Customization of the settings is possible, but requires in-depth knowledge and correct user rights settings. By default the content capture service IMiS®/Capture Service listens on the network interface "localhost" (127.0.0.1 or ::1) on port 5000/tcp, which enables local communication with the IMiS®/wScan application.
Additional user authentication is therefore not required.

Every access to the IMiS®/Capture Service must contain a special character string (a security key), which must be recorded in the IMiS®/Capture Service settings. A user with administrator authorization can enter the security key or create it via the IMiS®/wScan administrator module.

Access to the IMiS®/Capture Service is protected with the C.O.R.S. standard (https://en.wikipedia.org/wiki/Cross-origin_resource_sharing), which prevents Web browsers from accessing web domain services that are not specified in the IMiS®/Capture Service.
The Web browser will prevent access to the local IMiS®/Capture Service to any application on a different web domain, unless this domain is allowed in the IMiS®/Capture Service.
The user with administrative authorization can enter all the allowed web domains via the IMiS®/wScan administrator module.

A user logged in a Windows operating system can access services supported by IMiS®/Capture Service via the IMiS®/wScan application.
IMiS®/Capture Service runs in the system user context (SYSTEM account), which enables the user greater access to the operating system resources than a regular user with limited rights (restricted user). Certain resources are also available to a regular user only via IMiS®/Capture Service and the IMiS®/wScan application.

All IMiS®/Capture Service settings are stored in the Windows Registry or in a file system, which cannot be accessed by a user without administrator rights. The same applies to user-defined scanning profiles. Profile settings can be modified only with the IMiS®/wScan application or with the administrator module of IMiS®/Capture Service.

A user with administrator authorization can modify them outside of these two products, but has to have proper knowledge of modifying the Windows Registry. For more information see chapter 4.3.1 Additional administrator settings.

2.3 Functionalities

- Capture of contents and control of connected ISIS-compatible optical scanners.
- Capture of contents via various web browsers (e.g. Google Chrome, Mozilla Firefox, Microsoft Edge). A dedicated scanning application is therefore not required.
- The IMiS®/wScan application can be integrated into existing web applications.
- The entire capture of content and its processing is executed in the IMiS®/Capture Service backend service, which is based on Microsoft .NET technology.
- IMiS®/Capture Service contains modules for the capture and processing of content (barcode recognition, capture of metadata, etc.).
- IMiS®/Capture Service has been designed modularly, with each module responsible for its own phase of the capture or processing of content. This enables easy and quick upgrading with additional modules. By updating the profile, you can create your own module execution sequence.
- The IMiS®/wScan application has been designed in a JavaScript language without the use of additional technologies that browser makers consider unsuitable or dangerous (e.g. ActiveX plug-ins or access to “native” protocols NPAPI, COM, etc.).
- IMiS®/wScan libraries enable flexibility, customization, and an easy and quick development of your own web solutions using a JavaScript language.

2.3.1 Module for the capture of content from an optical scanner

IMiS®/Capture Service can capture content from all scanners supporting the ISIS industrial standard. This standard enables a wide range of functionalities and is supported by most scanner makers.

Standard scanner settings can be modified through the IMiS®/wScan application:

- Scanner selection.
- Scanning mode.
- Scanning resolution.
- Size of scanned page.

Additional settings that are specific to an individual scanner cannot be set through the IMiS®/wScan application, but only through the administrator module of IMiS®/Capture Service. For more information see Chapter 4.3.1 Additional administrator settings.

2.3.2 Module for saving content

IMiS®/Capture Service enables the saving of content to a file system.

Various file formats are available:

- BMP
- GIF
- TIFF
- JPEG
- PCX
- PDF/A
- PNG.

For each save format you can set the color and compression supported by the selected file format.

2.3.3 Module for barcode recognition

IMiS®/Capture Service enables the recognition of the following barcodes:

- 1D barcodes:
  Addon 2, Addon5, Australian Post, BDC Matrix, Codabar, Code-25 Datalogic, Code-25 IATA, Code-25 Industrial, Code-25 Interleaved, Code-25 Invert, Code-25 Matrix, Code-32, Code-39, Code-93, EAN-13, EAN-8, Type-128, UCC-128, UPC-A, UPC-E,
- 2D barcodes:
  AZTEC, Data Matrix, Intelligent Mail, PDF-417, Postnet, QR Code, Royal Post.

2.4 Application integration

The IMiS®/wScan application consists of three modules:

- **IMiS®/Capture Service**: a backend Windows service that executes the capture and processing of various contents.
- **imis.scan.js**: Javascript library that enables communication with IMiS®/Capture Service.
- **imis.scan.ui.js**: utility JavaScript library for displaying the already created visual components.

Each of the above-mentioned modules can be used for integration with another application.

2.4.1 Integration of IMiS®/Capture Service

Direct integration with IMiS®/Capture Service based on RESTful technology is not covered by this manual.

2.4.2 Integration of imis.scan.js library

The "imis.scan.js" library manages the exchange of data in JSON format with IMiS®/Capture Service using RESTful technology. Through it, a developer of applications in the JavaScript language can configure profiles or capture content (e.g. scan). It enables the detection of events during scanning, the reading of jobs, documents and pages. This library also serves as a basis for making your own web solutions. It does not need any other JavaScript libraries to work.
The developer must obtain a unique security key that is recorded in the IMiS®/Capture Service. For obtaining a security key see chapter 4.3.2 Security settings.

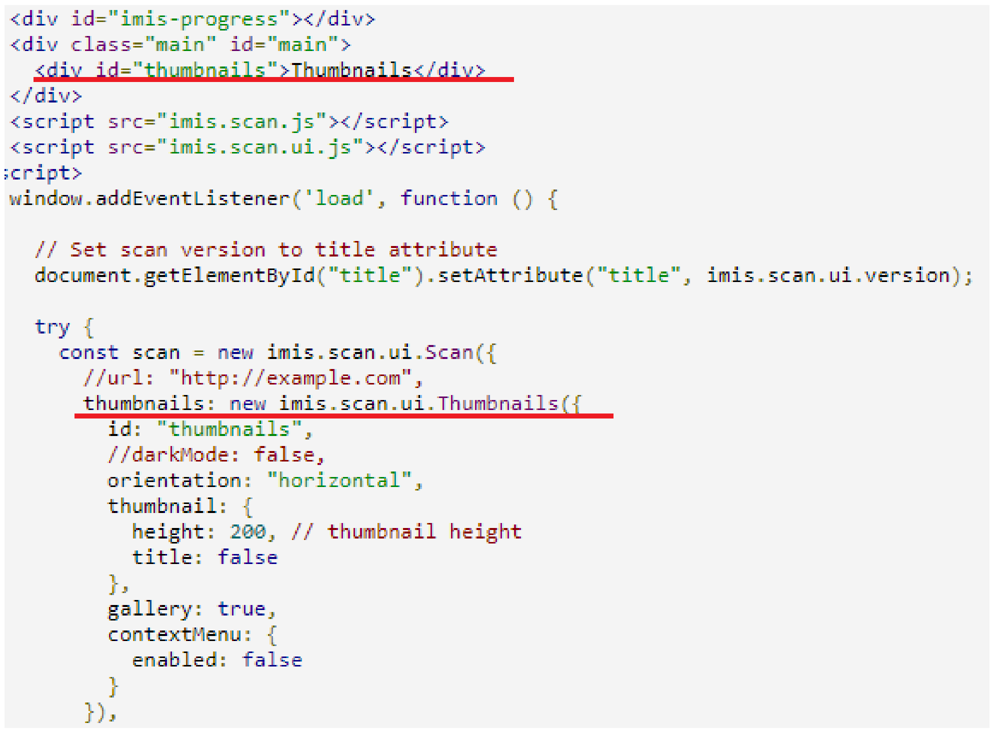
```
<!DOCTYPE html>
<html>
<head>
  <title>imis.scan.js</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="sample.css" />
</head>
<body class="sample">
  <h1>Sample</h1>
  <p>This example demonstrates reading scan profiles. Profiles are displayed in list.</p>

  <div>Profiles:</div>
  <ol id="profiles"></ol>
  <div id="error"></div>

  <script src="../imis.scan.js"></script>
  <script>
    window.addEventListener('load', function () {
      try {
        // Profiles ordered list
        var ol = document.getElementById("profiles");

        // Create a scan object
        var scan = new imis.scan.Scan();

        // Read profiles
        scan.getProfiles({
          success: function (profiles) {
            for (var i = 0; i < profiles.length; i++) {
              // Add profile to ordered list
              var li = document.createElement("li");
              li.innerHTML = profiles[i].name;
              ol.appendChild(li);
            }
          },
          error: function (error) {
            // Show error
            document.getElementById("error").innerHTML = error;
          }
        });
      } catch (e) {
        // Show error
        document.getElementById("error").innerHTML = e;
      }
    });
  </script>
</body>
</html>
```

Image 4: Example of using imis.scan.js library to read profiles

2.4.3 Integration of imis.scan.ui.js library

The “imis.scan.ui.js” library is intended for quicker and simpler development of your own solutions. It contains some of the most commonly used visual components (for executing, stopping and continuing jobs, selecting and setting profile properties, displaying job progress, displaying the selected page and its properties, and displaying all captured pages). A web application developer can easily build these components into the application and create a useful user interface without advanced knowledge of the HTML or CSS languages. All components come with specific settings, which can be used to change their appearance. All they need to operate is the imis.scan.js library and a unique security key, which is recorded in the IMiS®/Capture Service.

For obtaining a security key see chapter 4.3.2 Security settings.



```
<div id="imis-progress"></div>
<div class="main" id="main">
  <div id="thumbnails">Thumbnails</div>
</div>
<script src="imis.scan.js"></script>
<script src="imis.scan.ui.js"></script>
cript>
window.addEventListener('load', function () {

  // Set scan version to title attribute
  document.getElementById("title").setAttribute("title", imis.scan.ui.version);

  try {
    const scan = new imis.scan.ui.Scan({
      //url: "http://example.com",
      thumbnails: new imis.scan.ui.Thumbnails({
        id: "thumbnails",
        //darkMode: false,
        orientation: "horizontal",
        thumbnail: {
          height: 200, // thumbnail height
          title: false
        },
        gallery: true,
        contextMenu: {
          enabled: false
        }
      }),
```

Image 5: Example of using imis.scan.ui.js library to set the appearance of the Thumbnails component

2.5 Versions

Product version labeling is based on a scheme that includes the following:

- Three separate numerical identifiers (MAJOR, MINOR, RELEASE).
- An identifier of the 32-bit or 64-bit installation platform (PLATFORM).

The following is an example of a scheme:

IMiS.wScan.MAJOR.MINOR.RELEASE.PLATFORM.msi

The example of IMiS®/wScan installation package name:

IMiS.wScan.1.3.1802.x32.msi

The scheme consists of the name of the IMiS®/wScan module and the following elements:

- MAJOR: The identifier in the MAJOR position signifies the main/major version of the product. It is arbitrary and changes with regard to the volume of the changes and functionalities introduced. The identifier in this position changes the least.
  In the event that it is changed, it signifies a major difference in the product compared to the previously issued version (with a lower MAJOR version).
  This identifier has a range of values from 1-n; it is continuous and can only increase.
- MINOR: This identifier indicates a minor version of the product. It changes more frequently than the main version in terms of changes to the system, features and fixes.
  A change in the minor version represents smaller changes and fixes in the framework of the same product generation (indicated by the main or major version). The range of values is from 1 to n. This number is not continuous. It resets to its base value (1) with each new MAJOR version.
- RELEASE: This identifier represents the time component of the product release in accordance with the YYMM scheme.
  MM indicates the month of the release (range of values from 01 to 12), and YY indicates the last two digits of the year.

  Example: The RELEASE identifier for a product released in February of 2018 will read 1802.

- PLATFORM: Indicates the operating systems on which this application can be used.
  The 32-bit version can run on a 32-bit Windows operating system, and on a 64-bit one.
  The 64-bit version can run only on a 64-bit Windows operating system.

3 SYSTEM REQUIREMENTS

For successful installation and execution, the IMiS®/wScan application has the following hardware and software requirements.

3.1 Hardware

Practically all computers currently available on the market meet the hardware requirements for running IMiS®/wScan.
Minimum and recommended requirements are listed below.

3.1.1 Minimum requirements

Minimum requirements for IMiS®/wScan:

- Intel Core 2 Duo 2 GHz processor
- 1 GB RAM
- 150 MB of unused hard disk space
- TCP/IP network access (IPv4 or IPv6).

3.1.2 Recommended requirements

Recommended requirements for IMiS®/wScan:

- Intel Core i5 3 GHz processor or faster
- 2 GB RAM or more
- 250 MB of unused hard disk space
- TCP/IP network access (IPv4 or IPv6).

3.2 Software

Requirements for IMiS®/wScan:

- .NET 4.5
- Javascript ECMAScript 6
- Browsers with enabled WebSocket technology and ECMAScript6 standard support:
  - Google Chrome: minimum version 50
  - Mozilla Firefox: minimum version 45
  - Microsoft Edge: minimum version 20.
- Supported operating systems:
  - Windows 10; Windows 8.x and Windows 7 SP1.

4 PRODUCT MANAGEMENT

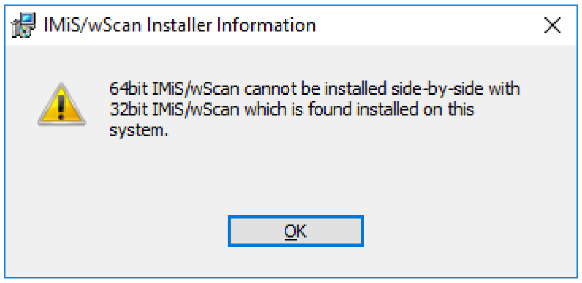
The IMiS®/wScan application is managed by administrators and/or application developers. Management includes installation, startup, closing, upgrading and uninstallation.

4.1 Installation

The installation can be performed in an environment that meets at least the minimum installation requirements. IMiS®/wScan installation can be performed with a setup wizard as administrative installation or "silent" installation. In any case, the displayed notifications and dialog boxes are in English.

Warning:
Prior to installation stop the IMiS®/Scan application, because IMiS®/wScan does not work properly if the IMiS®/Scan application is running.

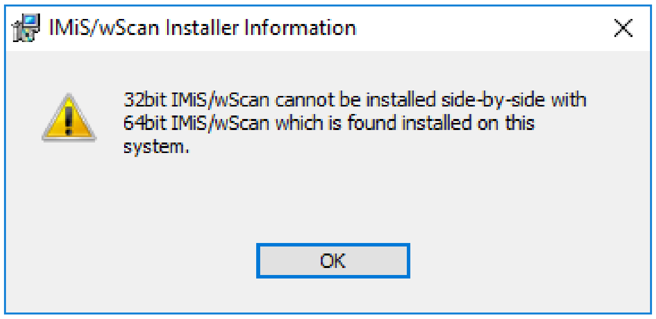
Warning:
Installation of a 64-bit version of IMiS®/wScan will be unsuccessful if the 32-bit version is already installed on the workstation.


IMiS/wScan Installer Information

64bit IMiS/wScan cannot be installed side-by-side with 32bit IMiS/wScan which is found installed on this system.

Image 6: Notification of prohibited simultaneous installation of the 64-bit and 32-bit version of IMiS®/wScan

The same applies vice-versa. If the 64-bit version of IMiS®/wScan is already installed on the workstation, installation of a 32-bit version will be unsuccessful.

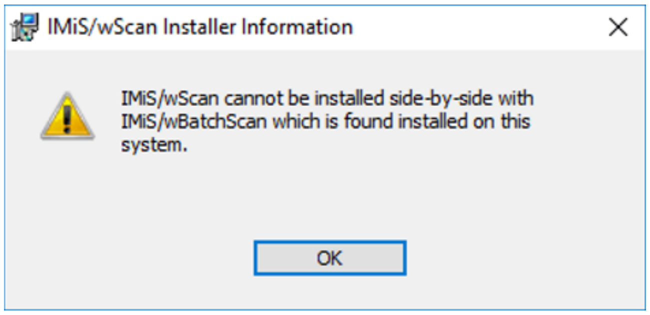

IMiS/wScan Installer Information

32bit IMiS/wScan cannot be installed side-by-side with 64bit IMiS/wScan which is found installed on this system.

Image 7: Notification of prohibited simultaneous installation of the 32-bit and 64-bit version of IMiS®/wScan

<u>Warning:</u>

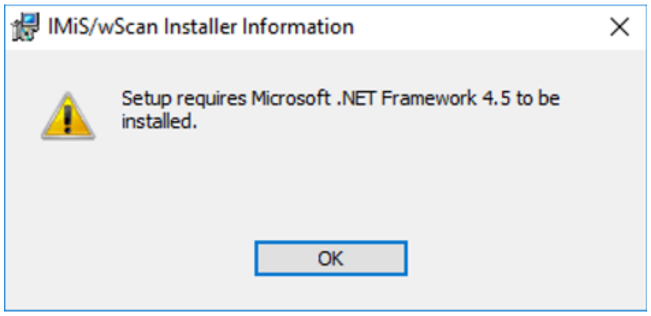
Installation of the IMiS®/wScan application is not possible on a workstation on which IMiS®/wBatchScan has already been installed.


IMiS/wScan Installer Information

IMiS/wScan cannot be installed side-by-side with IMiS/wBatchScan which is found installed on this system.

Image 8: Notification of prohibited simultaneous installation of IMiS®/wScan and IMiS®/wBatchScan

<u>Warning:</u>
Installation of the IMiS®/wScan application on a workstation is not possible if .NET Framework 4.5 has not been installed.


IMiS/wScan Installer Information

Setup requires Microsoft .NET Framework 4.5 to be installed.

Image 9: Notification of required installation of .NET Framework 4.5

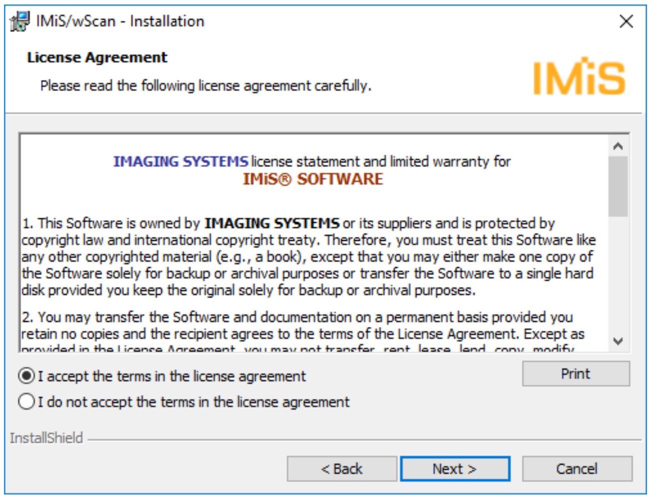
4.1.1 Installation with the Wizard

The user interface of the installation package guides the administrator through the installation procedure.
The administrator installs the IMiS®/wScan application on a workstation in a Windows environment with one or several optical scanners physically connected. This application includes the IMiS®/Capture Service web service and the relevant libraries.

<u>Example</u> of the name of an installation package:
IMiS.wScan.1.3.1802.x64.msi

Installation begins with launching the installation package from the file system. A dialog box appears, informing the administrator that the installation package is preparing for installation.

In the next step read the terms of the license agreement carefully. If you agree with them, select "I accept the terms in the license agreement" and thus fully accept the license terms.
If you do not agree with the license terms, select "I do not accept terms in the license agreement" and by clicking on the "Cancel" button cancel the installation procedure.


IMiS/wScan - Installation

License Agreement

Please read the following license agreement carefully.

IMAGING SYSTEMS license statement and limited warranty for
IMiS® SOFTWARE

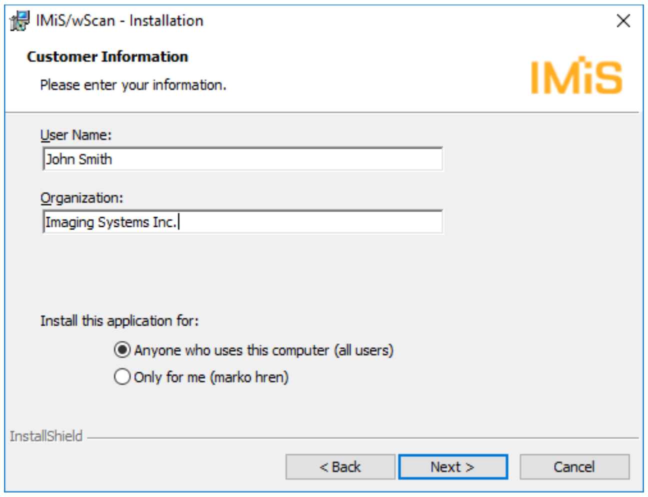
1. This Software is owned by IMAGING SYSTEMS or its suppliers and is protected by copyright law and international copyright treaty. Therefore, you must treat this Software like any other copyrighted material (e.g., a book), except that you may either make one copy of the Software solely for backup or archival purposes or transfer the Software to a single hard disk provided you keep the original solely for backup or archival purposes.

2. You may transfer the Software and documentation on a permanent basis provided you retain no copies and the recipient agrees to the terms of the License Agreement. Except as

[✓ I accept the terms in the license agreement]
I do not accept the terms in the license agreement

Image 10: Viewing and accepting the license terms

The installation procedure is continued by entering the user name into the input field “User Name” and the organization into the input field “Organization”. Select whether the application will be installed only for the current user ("Only for me") or for all users on this computer ("Anyone who uses this computer").


IMiS/wScan - Installation

Customer Information

Please enter your information.

User Name: John Smith

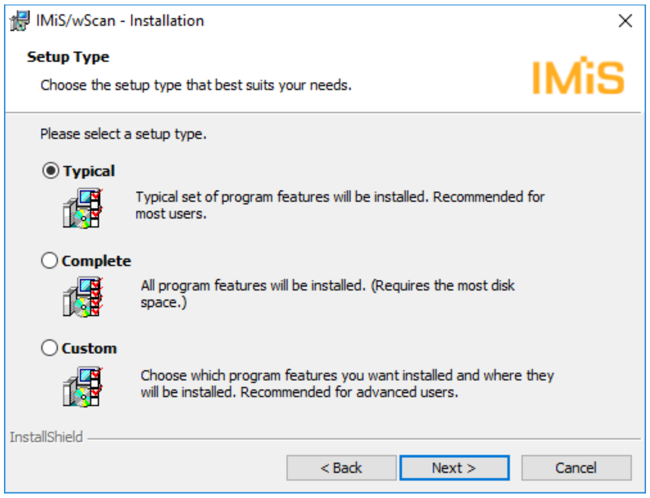
Organization: Imaging Systems Inc.

Install this application for:

[✓ Anyone who uses this computer (all users)]
Only for me (marko hren)

Image 11: Entering data on application user

In the next step select between typical installation ("Typical"), full installation ("Complete") or user-customized installation ("Custom").


IMiS/wScan - Installation

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

(●) **Typical**
Typical set of program features will be installed. Recommended for most users.

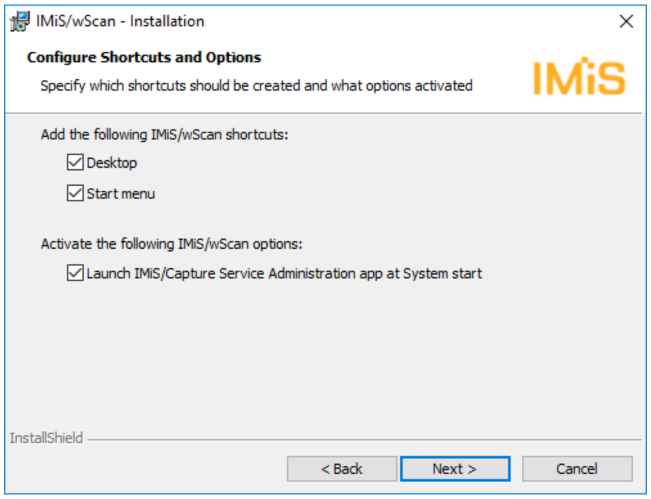
( ) **Complete**
All program features will be installed. (Requires the most disk space.)

( ) **Custom**
Choose which program features you want installed and where they will be installed. Recommended for advanced users.

InstallShield

Image 12: Selecting between typical, full or user-customized installation

For all installation types the administrator specifies which shortcuts will be created and which options will be activated during the installation procedure.


IMiS/wScan - Installation

Configure Shortcuts and Options

Specify which shortcuts should be created and what options activated

Add the following IMiS/wScan shortcuts:

[✓] Desktop
[✓] Start menu

Activate the following IMiS/wScan options:

[✓] Launch IMiS/Capture Service Administration app at System start

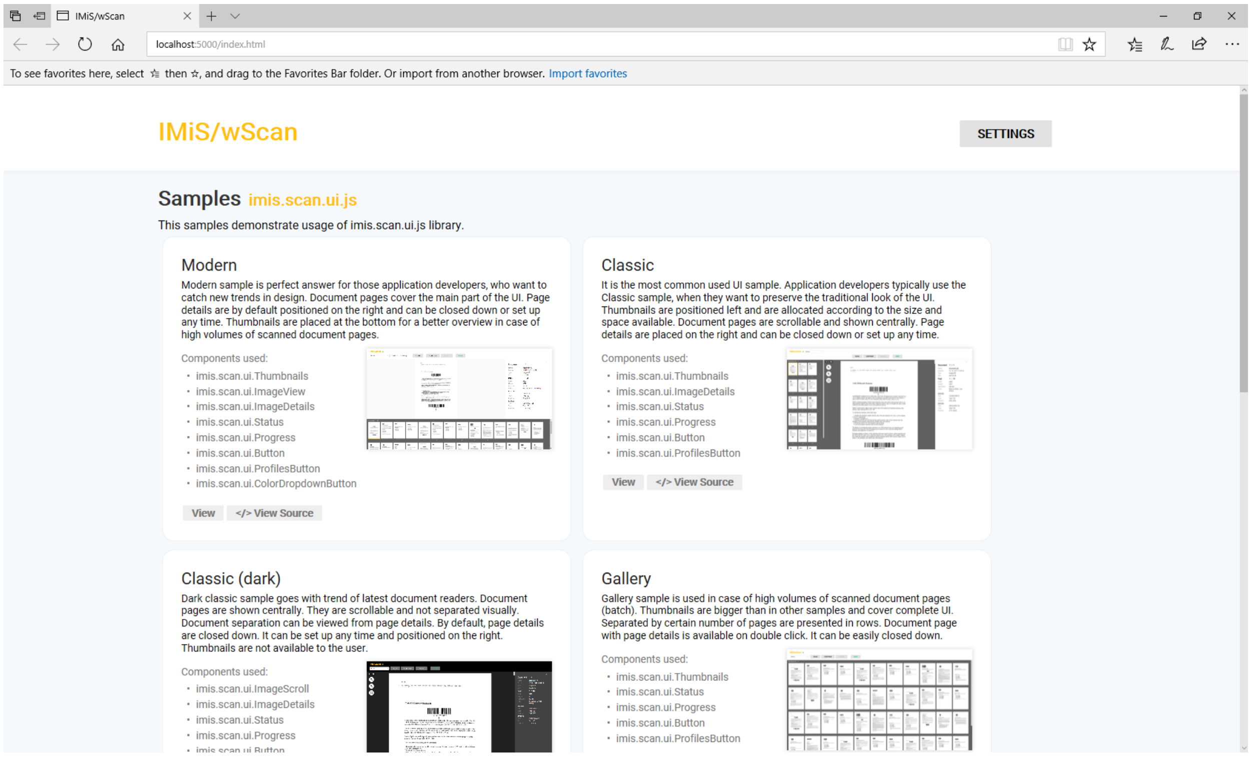
InstallShield

Image 13: Selecting which shortcuts will be created and which options will be activated during installation

If the administrator ticks the choice “Launch IMiS®/Capture Service Administration app at System start”, the administration module will launch when the workstation starts.

If you tick the choice “Desktop”, shortcuts to the IMiS®/wScan start page and the IMiS®/Capture Service administration module will be installed on the desktop.

If you tick the choice "Start menu", a shortcut to the IMiS®/wScan start page and the IMiS®/Capture Service administration module will be added to the Start menu.


IMiS/wScan
localhost:5000/index.html
To see favorites here, select ☆ then ☆, and drag to the Favorites Bar folder. Or import from another browser. Import favorites
IMiS/wScan
SETTINGS

Samples imis.scan.ui.js

This samples demonstrate usage of imis.scan.ui.js library.

Modern

Modern sample is perfect answer for those application developers, who want to catch new trends in design. Document pages cover the main part of the UI. Page details are by default positioned on the right and can be closed down or set up any time. Thumbnails are placed at the bottom for a better overview in case of high volumes of scanned document pages.

Components used:

- imis.scan.ui.Thumbnails
- imis.scan.ui.ImageView
- imis.scan.ui.ImageDetails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton
- imis.scan.ui.ColorDropdownButton

View </> View Source

Classic

It is the most common used UI sample. Application developers typically use the Classic sample, when they want to preserve the traditional look of the UI. Thumbnails are positioned left and are allocated according to the size and space available. Document pages are scrollable and shown centrally. Page details are placed on the right and can be closed down or set up any time.

Components used:

- imis.scan.ui.Thumbnails
- imis.scan.ui.ImageDetails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton

View </> View Source

Classic (dark)

Dark classic sample goes with trend of latest document readers. Document pages are shown centrally. They are scrollable and not separated visually. Document separation can be viewed from page details. By default, page details are closed down. It can be set up any time and positioned on the right. Thumbnails are not available to the user.

Components used:

- imis.scan.ui.ImageScroll
- imis.scan.ui.ImageDetails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button

Gallery

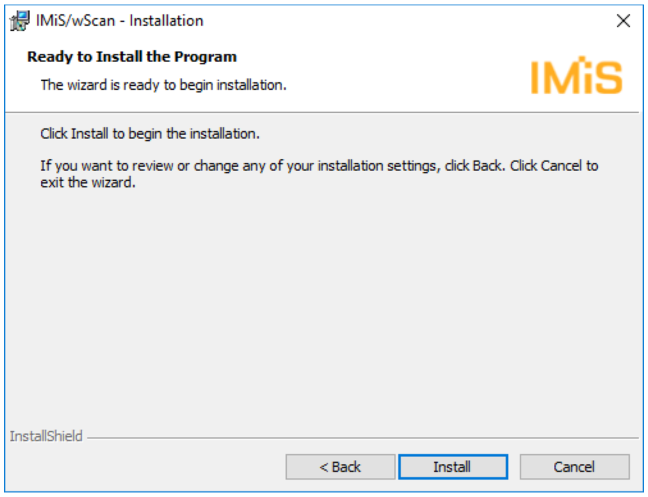
Gallery sample is used in case of high volumes of scanned document pages (batch). Thumbnails are bigger than in other samples and cover complete UI. Separated by certain number of pages are presented in rows. Document page with page details is available on double click. It can be easily closed down.

Components used:

- imis.scan.ui.Thumbnails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton

Image 14: Displaying the result of selecting "Start menu"

Typical installation, which is recommended for most users, begins by transferring predefined files to the file system. The administrator confirms the selected installation setting and starts the installation procedure by clicking on the "Install" button.


IMiS/wScan - Installation

Ready to Install the Program

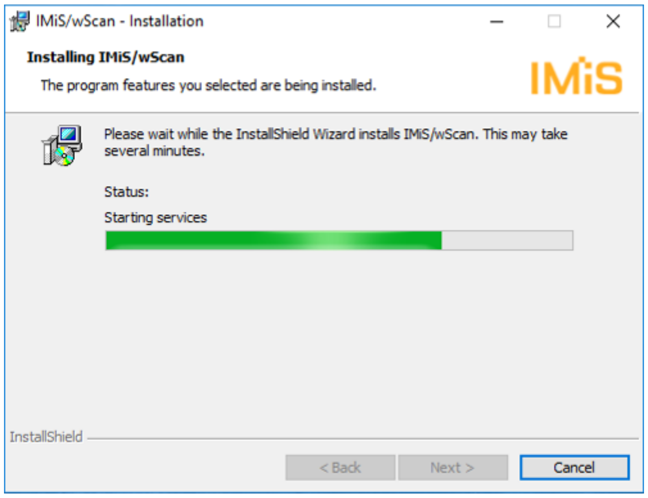
The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Image 15: Starting the installation procedure

The installation procedure for the IMiS®/wScan software product begins; the progress bar shows information about the transfer of files to the appropriate locations. Installation may take a few tens of seconds, depending on the version of the installation package and computer speed.


IMiS/wScan - Installation

Installing IMiS/wScan

The program features you selected are being installed.

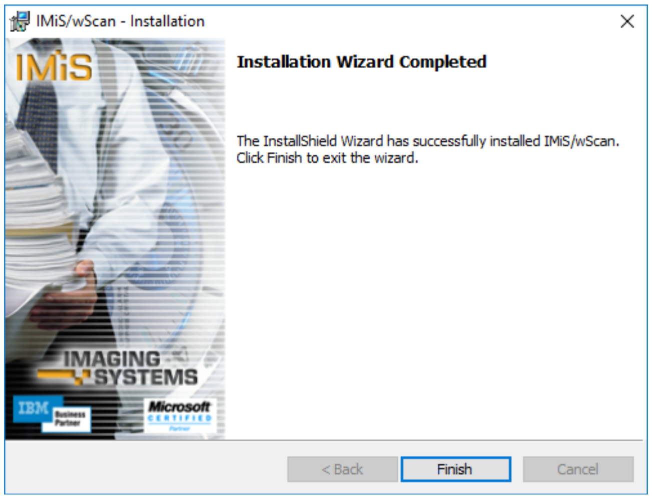
Please wait while the InstallShield Wizard installs IMiS/wScan. This may take several minutes.

Status:

Starting services

Image 16: Displaying the progress bar during the installation procedure

Installation is complete when the last display box appears, which you close by clicking on the "Finish" button.



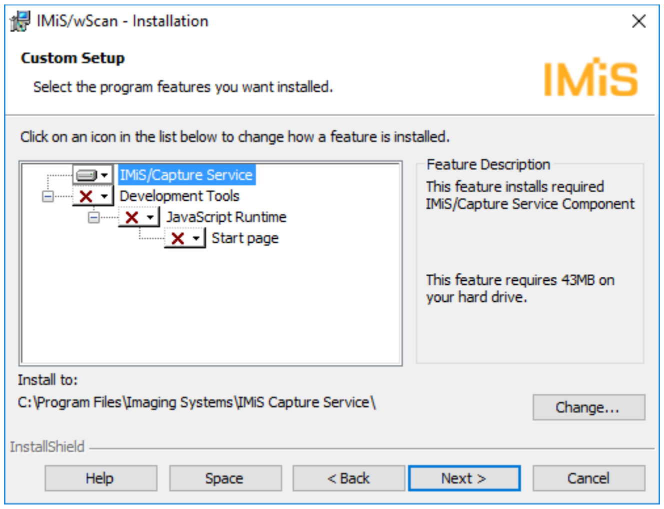
Installation Wizard Completed

The InstallShield Wizard has successfully installed IMiS/wScan. Click Finish to exit the wizard.

Image 17: Notification of completing the installation procedure

The same procedure as for typical installation is carried out during full installation.
Full installation will install all of the features from the installation package in the file system, which is why it requires the most disk space.

User-customized installation ("Custom") will install only specific features in the file system. It is intended for advanced users.



Custom Setup

Select the program features you want installed.

Click on an icon in the list below to change how a feature is installed.

Feature Description

This feature installs required IMiS/Capture Service Component

This feature requires 43MB on your hard drive.

Install to:

C:\Program Files\Imaging Systems\IMiS Capture Service\

Image 18: Selecting the features of application installation

The administrator confirms the selected installation setting and starts the installation procedure by clicking on the "Install" button. Further steps are the same as for typical and full installation.

4.1.2 Silent installation

The IMiS®/wScan application can also be installed without user control. Installation is performed silently, without displaying the user interface. The installation is executed via the *msiexec.exe* utility program. This tool is a part of the Microsoft installation product and is used for performing various types of maintenance in applications that are installed in the Windows operating system.

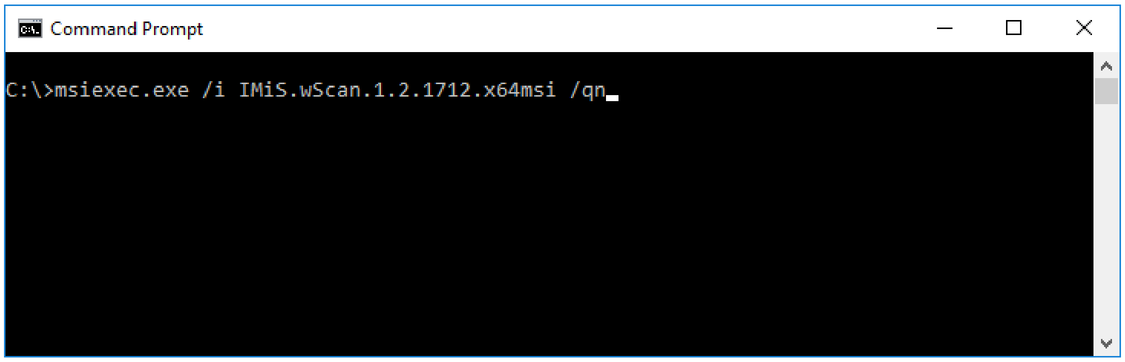
For a full list of the supported functions of the *msiexec.exe* program, turn to the Microsoft article collection: http://msdn.microsoft.com/en-us/library/windows/desktop/aa367449(v=vs.85).aspx
The utility program is executed from the command bar.

For a list of all parameters, turn to the Microsoft website:
http://msdn.microsoft.com/en-us/library/windows/desktop/aa367988(v=vs.85).aspx.
Installation may take a few tens of seconds, depending on computer speed.

Example of a command bar for silent installation:

C:\Windows\system32\msiexec.exe /i IMiS.wScan.1.3.1802.x64.msi /qn

Below is a command bar for the silent installation of IMiS®/wScan:



```
C:\>msiexec.exe /i IMiS.wScan.1.2.1712.x64msi /qn
```

Image 19: Example of a command bar for silent installation in previous version

The table below lists the various silent installation modes:

Command bar parameters	**Description**
/q, /qn	No user interface.
/qn+	No user interface with a modal window at the end of installation.
/qb	Basic user interface with a simple progress bar. The "/gb!" parameter is used to hide the "Cancel" button.
/qr	Simplified user interface without a modal window at the end of installation.
/qf	A full user interface with all the display boxes, a progress bar and error message at the end of installation.

Image 20: Silent installation settings

Before launching the installation of the IMiS®/wScan application, you can specify various parameters that are specific to this installation. Add them to the end of the command bar using the syntax:

c:\windows\system32\msiexec.exe /i IMiS.wScan.1.3.1802.x64.msi /qn PARAMETER=VALUE

The table below describes the supported command bar parameters:

PARAMETER	**Valid values**	**Description**
INSTALLDIR	<directory name>	This parameter contains the default destination folder for installation files (default value = "%PROGRAMFILES%\Imaging Systems\IMiS Capture Service\").
USERNAME	<username>	This parameter contains the username of the user performing the installation (default value is taken from system settings).
COMPANYNAME	<company name>	This parameter adds the company name to the installation (default value is taken from system settings).

PARAMETER	Valid values	Description
SHORTCUT_START	1 / 0	This parameter is used to inform the installation process to create a shortcut in the "Programs" menu (Default value is 1).
SHORTCUT_DESKTOP	1 / 0	This parameter is used to inform the installation process to create a desktop shortcut (Default value is 1).
LAUNCH_ADMIN_ON_START	1 / 0	This parameter is used to inform the installation process to create the necessary registry records and enable automatic launch of the IMiS®/Capture Service administration module when the workstation starts. (Default value is 1)
ADDLOCAL	ALL	Enables the silent installation of all components of the installation package, which is equivalent to selecting "Complete" in the installation with the Wizard.

Image 21: Supported command bar parameters

4.2 Startup and closing

IMiS®/Capture Service launches automatically when the workstation connected to the optical scanner starts.

Starting and stopping IMiS®/Capture Service is also possible manually by double-clicking on the IMiS®/wScan administration module. After startup the IMiS® icon appears at the bottom of the desktop.

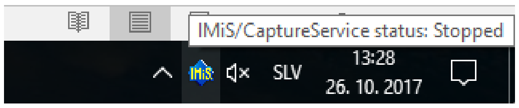

IMiS/CaptureService status: Stopped

Image 22: Displaying the current status of IMiS®/Capture Service: stopped

By right-clicking on the IMiS® icon, the menu appears. By selecting the option “Start service”, the administrator starts IMiS®/Capture Service.

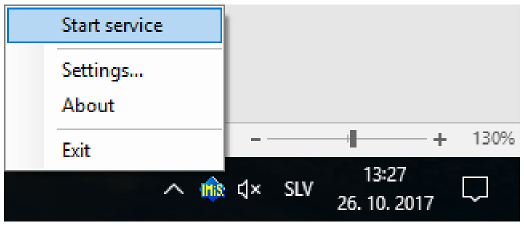


Image 23: Selecting the option to start IMiS®/Capture Service

It takes a few seconds for the service to start, as it initializes properly and checks the suitability of the scanner driver. The current status of the service is visible if you move the mouse over the IMiS® icon.

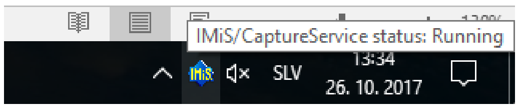


Image 24: Displaying the status of IMiS®/Capture Service: running

The service is stopped by selecting the option “Stop service” in the menu of the IMiS® icon.

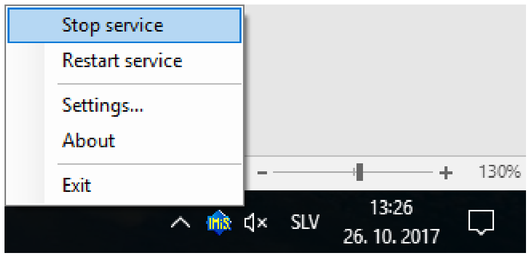


Image 25: Selecting the option to stop IMiS®/Capture Service

If you (the administrator) wish to restart the service, select the option “Restart service” in the menu of the IMiS® icon.

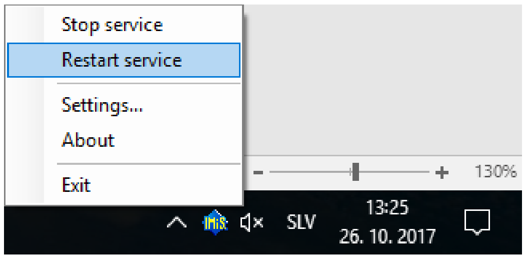


Image 26: Selecting the option to restart IMiS®/Capture Service

<u>Warning:</u> The administrator must refresh IMiS®/wScan application in browser (MS Edge, Mozilla Firefox, Google Chrome, ...), after restarting IMiS®/CaptureService.

4.3 Additional settings

Not all IMiS®/wScan application settings can be implemented via the "imis.scan.js" JavaScript library. The administrator can access additional settings by double-clicking on the IMiS®/wScan administration module. After startup the IMiS® icon appears at the bottom of the desktop.
By right-clicking on the IMiS® icon, the menu appears.

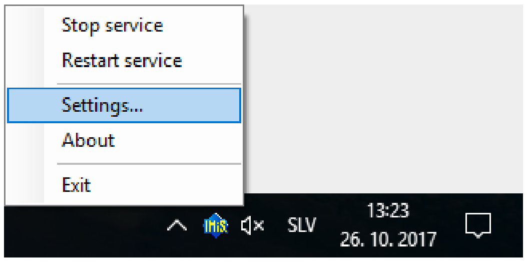
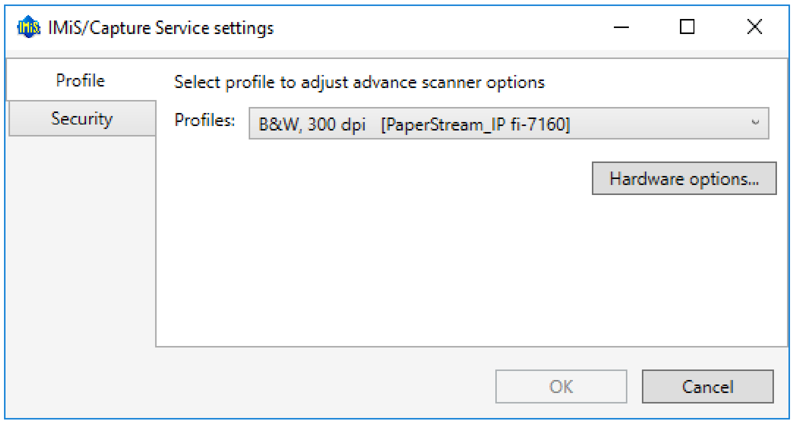


Image 27: Selecting the option to show additional settings

By selecting the option “Settings”, the configuration dialog box appears.



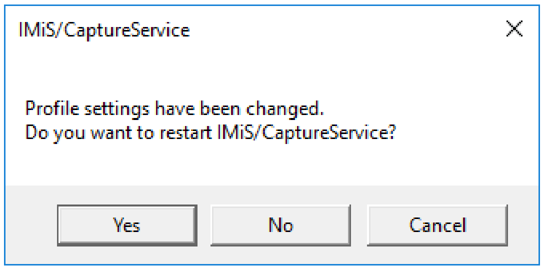
Select profile to adjust advance scanner options

Profiles: B&W, 300 dpi [PaperStream_IP fi-7160]

Image 28: Dialog box for setting additional settings

Tabs for additional settings for profiles and security are available. After choosing each of the tabs, additional settings are displayed. After the settings are set, the user with administrator authorization clicks the »OK button«.

If the profile has been modified and IMiS®/Capture Service has not been started, the additional settings will be saved. They will be used at the next startup of IMiS®/Capture Service.
If IMiS®/Capture Service has been started, a dialog box appears for restarting IMiS®/Capture Service.



Profile settings have been changed.
Do you want to restart IMiS/CaptureService?

Image 29: Dialog box for restarting IMiS®/Capture Service

Clicking on the “Yes” button starts the process of restarting IMiS®/Capture Service.
The configuration dialog box closes in the process.

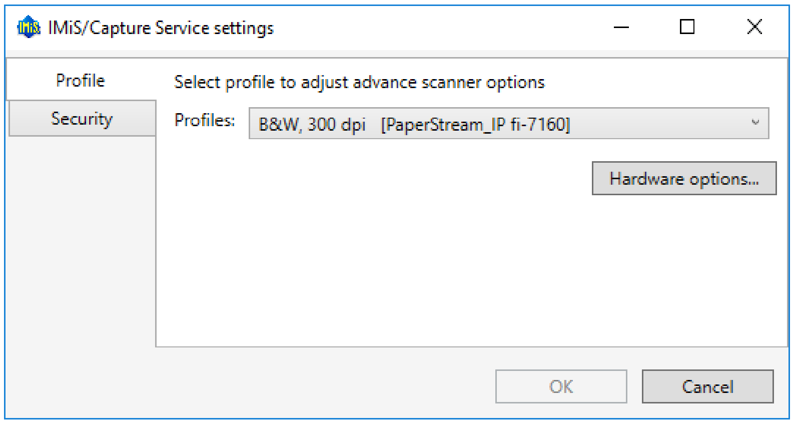
By clicking on the “No” button, the changed settings are saved. They will be applied at the next startup of IMiS®/Capture Service. The configuration dialog box closes.
By clicking on the “Cancel” button, the changes are not saved. The configuration dialog box stays open.

If the administrator selects the “Cancel” button in the configuration dialog box, the box closes without saving the changes made to the profile.

Warning: The administrator must refresh IMiS®/wScan application in browser (MS Edge, Mozilla Firefox, Google Chrome, ...), after restarting IMiS®/CaptureService.

4.3.1 Additional profile settings

For additional profile settings, the user with administrator authorization selects the »Profiles« tab. Settings are displayed on the right side. In the »Profiles« dropdown menu all the specified profiles are available. Besides the name of the profile, the scanner model is also displayed.

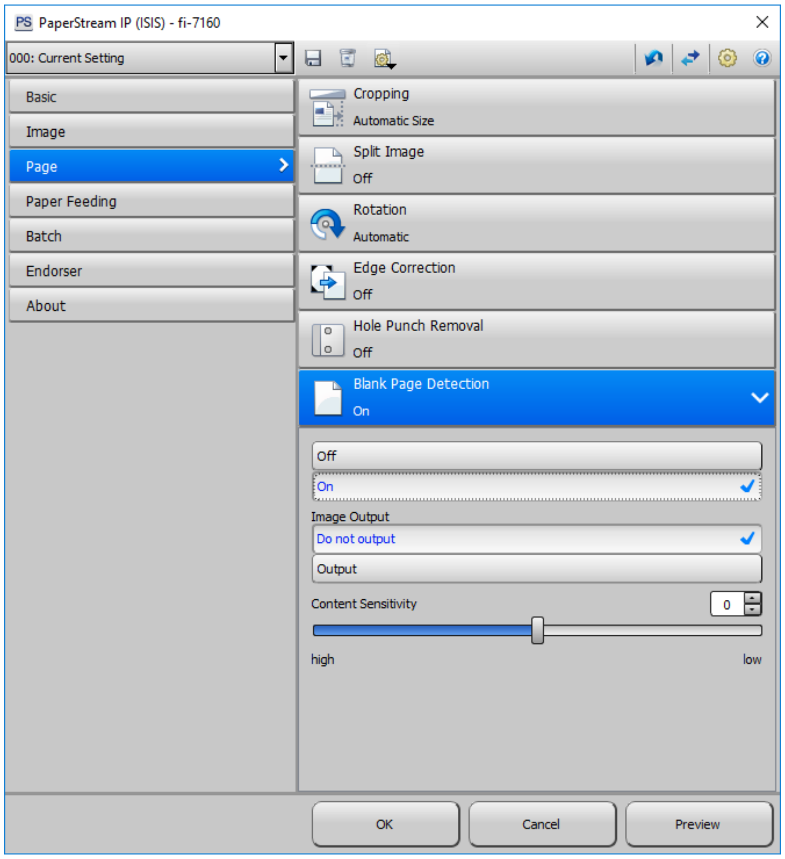

IMiS/Capture Service settings

Select profile to adjust advance scanner options

Profiles: B&W, 300 dpi [PaperStream_IP fi-7160]

Image 30: Dialog box for profile and security settings

The administrator selects the profile in which to set additional scanner settings (e.g. remove empty pages). After selecting the profile, click on the button “Hardware options...”. The scanner’s configuration dialog box appears.



PaperStream IP (ISIS) - fi-7160

000: Current Setting

Cropping: Automatic Size
Split Image: Off
Rotation: Automatic
Edge Correction: Off
Hole Punch Removal: Off
Blank Page Detection: On

Off
[✓ On]

Image Output
[✓ Do not output]
Output

Content Sensitivity: 0
high low

Image 31: Configuration dialog box of Fujitsu PaperStream driver

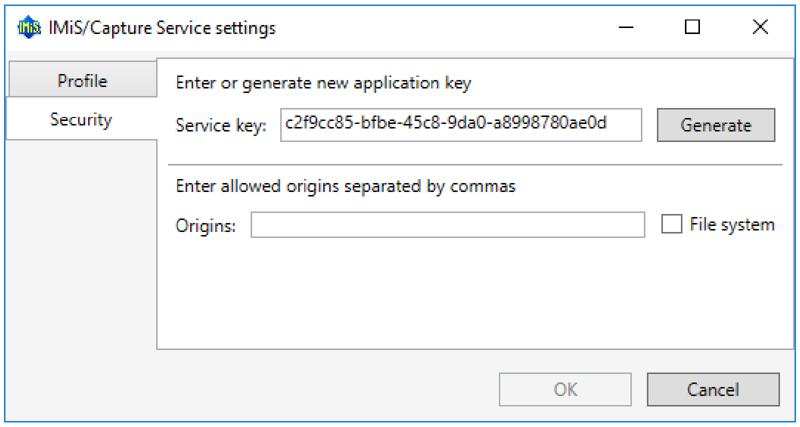
<u>Note</u>: Scanner makers use different configuration dialog boxes.

After finishing the settings, click on the “OK” button.

4.3.2 Security settings

A user with administrator authorization selects the »Security« tab.

Security settings that include a field for entering or obtaining a security key and entering the allowed indirect web domains is displayed.


IMiS/Capture Service settings

Enter or generate new application key

Service key: c2f9cc85-bfbe-45c8-9da0-a8998780ae0d

Enter allowed origins separated by commas

Origins:

File system

Image 32: Dialog box for security settings

A user with administrator authorization can enter any character string in the »Service key« field or click the »Generate« button to obtain a new unique character string. The developer of the web application enters the security key in his application by forwarding this character string to the **imis.scan.js** javascript library**.**

For more information see chapter 2.4.2 Integration of imis.scan.js library.

The IMiS®/Capture Service will reject all REST requests by web applications that don't contain the same security key as entered in the »Service key« field.

A user with administrator authorization enters all web domains from which the web application can execute REST requests to the IMiS®/Capture Service in the »Origins« field.

Individually entered web domains are separated by commas. If the »Origins« field is empty, only access via the »localhost« local network interface is allowed.

For access from all web domains, the administrator must enter the »*« sign (asterisk).

If the web application is executed directly from the file system, a user with administrator authorization must tick the »File system« field.

Warning: The administrator must refresh IMiS®/wScan application in browser (MS Edge, Mozilla Firefox, Google Chrome, ...), after restarting IMiS®/CaptureService.

4.3.3 Additional administrator settings

Profiles can also be configured outside of the IMiS®/wScan application. This can be done by a user with administrator rights and knowledge of updating Windows Registry.
All profile settings are written in Windows Registry under the key *HKEY_LOCAL_MACHINE\SOFTWARE\Imaging Systems\IMiS Capture Service* in the *profiles* field. If IMiS®/Capture Service does not have access rights for this key, the settings are saved to the file system in the *profiles.json* file in the *C:\ProgramData\Imaging Systems\IMiS Capture Service* folder.
The profile settings are written in the JSON file format. They can therefore be copied from one computer to another.

4.4 Uninstallation and modification

Installation modification or uninstallation of the IMiS®/wScan application is performed by the administrator on the workstation using the standard Windows application "Add or Remove Programs". You access the application by clicking on the "Start" button, locating the "Settings" icon and starting "Add or Remove Programs". From the Apps & features list the administrator selects the IMiS®/wScan application.

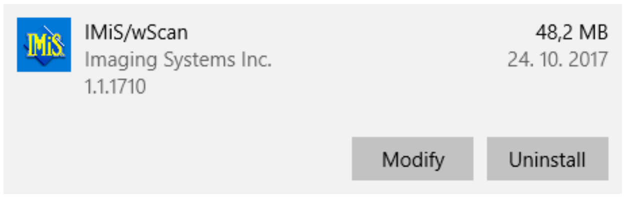

IMiS/wScan 48,2 MB
Imaging Systems Inc. 24. 10. 2017
1.1.1710

Image 33: Selecting between installation modification and uninstallation of application

4.4.1 Uninstall

By selecting the "Uninstall" option, the administrator begins the procedure of uninstalling the IMiS®/wScan application.

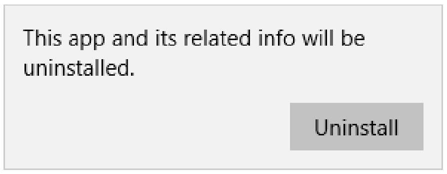
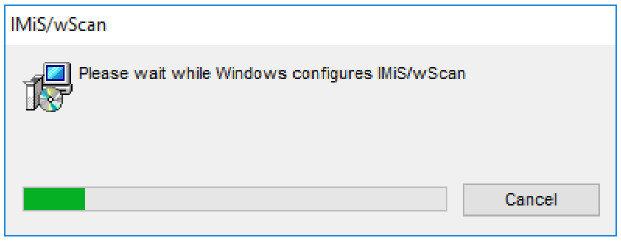

This app and its related info will be uninstalled.

Image 34: Selecting the uninstallation of application

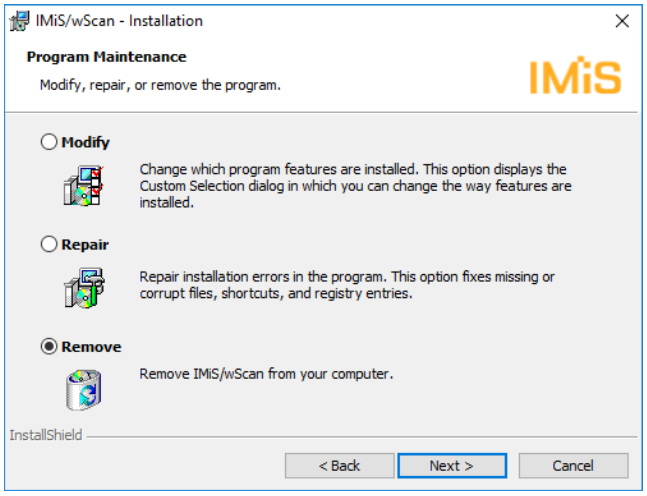
During the uninstallation procedure all files and application settings created by the installation package are removed. The administrator can monitor the progress of the configuration review via a dialog box. By clicking on the "Cancel" button, the review procedure is canceled.


IMiS/wScan

Please wait while Windows configures IMiS/wScan

Image 35: Displaying the progress bar of the configuration review

Afterwards the administrator is shown a dialog box for selecting the options: "Modify", "Repair" or "Remove". To remove the installation package, select "Remove". Confirm the selection with the "Next" button.


IMiS/wScan - Installation

Program Maintenance

Modify, repair, or remove the program.

Modify

Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

Repair

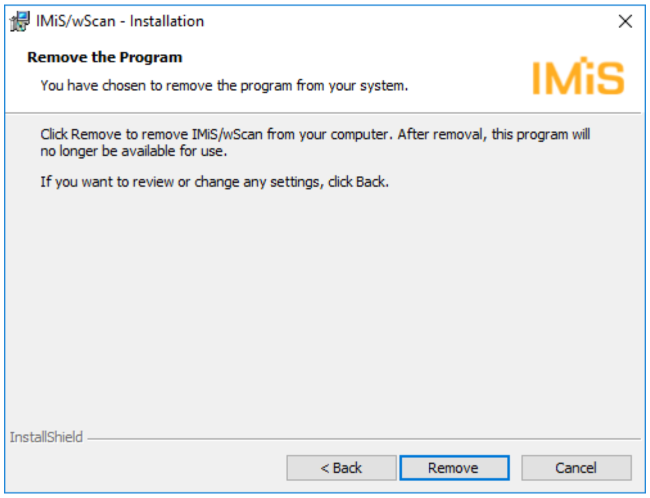
Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

● Remove

Remove IMiS/wScan from your computer.

Image 36: Selecting application removal

In the next step confirm the removal by clicking on the “Remove” button.


IMiS/wScan - Installation

Remove the Program

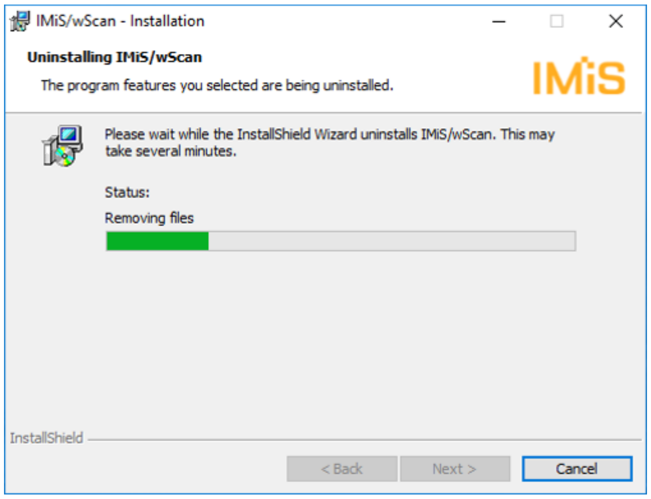
You have chosen to remove the program from your system.

Click Remove to remove IMiS/wScan from your computer. After removal, this program will no longer be available for use.

If you want to review or change any settings, click Back.

Image 37: Confirming application removal

Uninstallation may take from a few seconds to a few tens of seconds, depending on the version of the installation package and computer speed.


IMiS/wScan - Installation

Uninstalling IMiS/wScan

The program features you selected are being uninstalled.

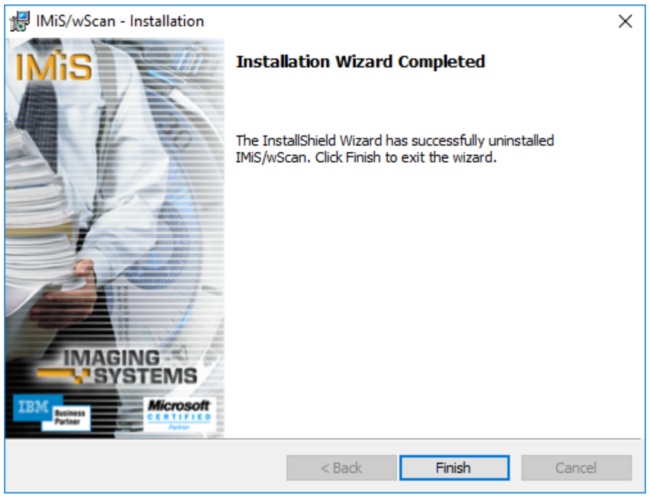
Please wait while the InstallShield Wizard uninstalls IMiS/wScan. This may take several minutes.

Status:

Removing files

Image 38: Displaying the progress bar during the uninstallation procedure

After finishing uninstalling the application, a dialog box appears, which the administrator closes by clicking on the “*Finish*” button.



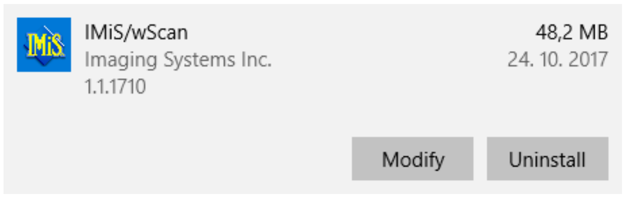
Installation Wizard Completed

The InstallShield Wizard has successfully uninstalled IMiS/wScan. Click Finish to exit the wizard.

Image 39: Notification of finishing the procedure of uninstalling the installation package

4.4.2 Installation modifications and repairs

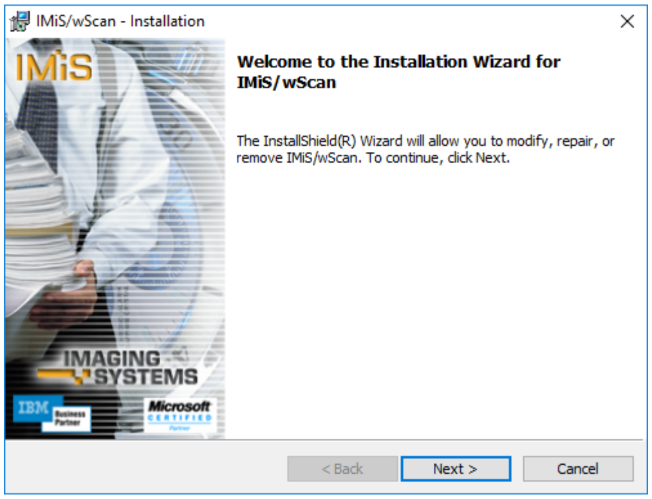
The administrator makes modifications and repairs to the installation of the IMiS®/wScan application in a Windows environment via the “Start” button, the “Settings” icon, “Add or Remove Programs”, and an application selected from the Apps & features list.



IMiS/wScan 48,2 MB
Imaging Systems Inc. 24. 10. 2017
1.1.1710

Image 40: Selecting between installation modifications and repairs and removing the installed application

The administrator confirms the selection of installation modifications or repairs by clicking on the “Next” button.


IMiS/wScan - Installation

Welcome to the Installation Wizard for IMiS/wScan

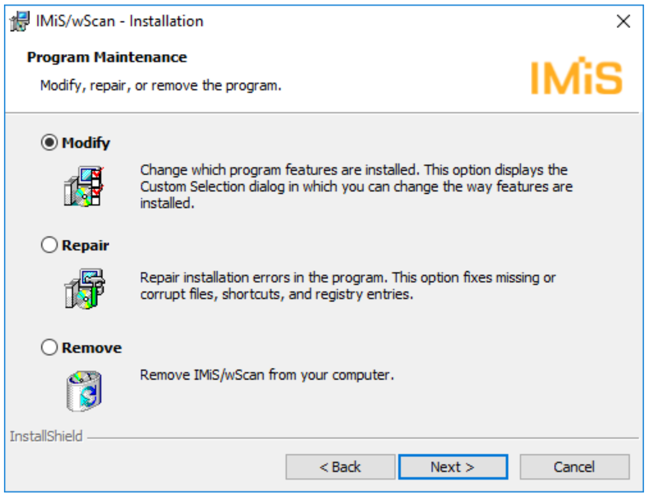
The InstallShield(R) Wizard will allow you to modify, repair, or remove IMiS/wScan. To continue, click Next.

Image 41: Starting the procedure of implementing installation modifications and repairs

4.4.2.1 Installation modifications

The administrator is shown a dialog box in which the option “Modify” has been ticked.

Confirm the selection with the “Next” button.


IMiS/wScan - Installation

Program Maintenance

Modify, repair, or remove the program.

[✓ **Modify**]

Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

Repair

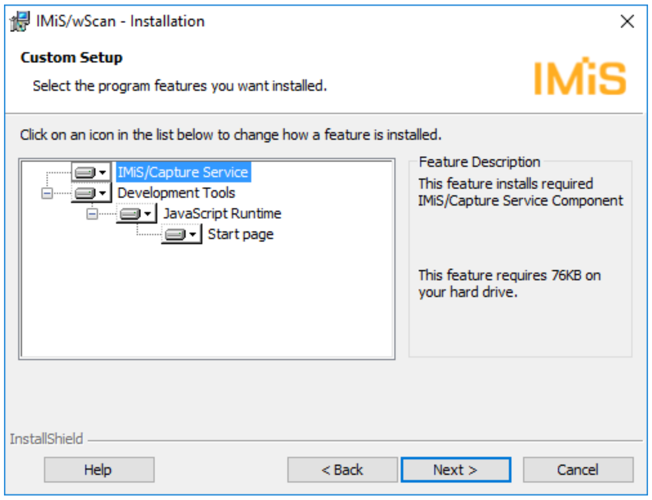
Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

Remove

Remove IMiS/wScan from your computer.

Image 42: Selecting installation modification

By clicking on the icon, the administrator ticks the application features to install.


IMiS/wScan - Installation
Custom Setup
Select the program features you want installed.
IMiS
Click on an icon in the list below to change how a feature is installed.
IMiS/Capture Service
Development Tools
JavaScript Runtime
Start page
Feature Description
InstallShield
Help
< Back
Next >
Cancel

This feature installs required IMiS/Capture Service Component

This feature requires 76KB on your hard drive.

Image 43: Selecting features when modifying installation

By confirming the selection, the administrator starts the installation procedure. Further steps are the same as for typical, full and custom installation. The procedure finishes with the installation of all required application features.

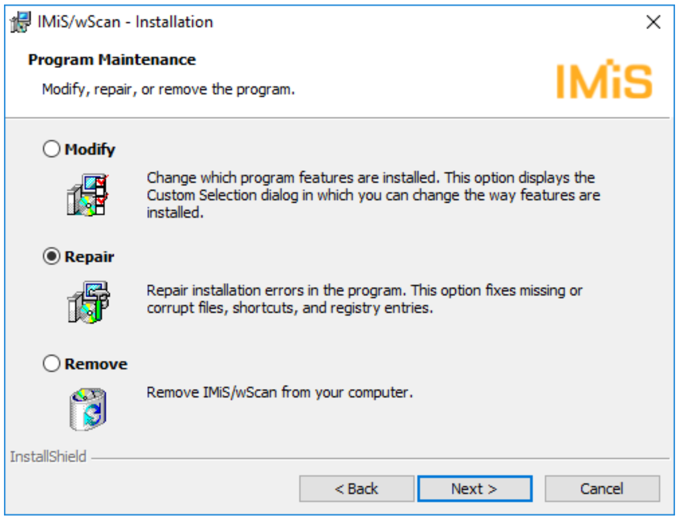
For more information see chapter 4.1.1 Installation with the Wizard.

4.4.2.2 Installation repairs

If installation files, shortcuts or register entries were damaged during the installation of the IMiS®/wScan application or later, the administrator can repair them.

After starting the procedure of making modifications and repairs to the installation, the administrator is shown a dialog box in which the option "Repair" has been ticked.

Confirm the selection with the "Next" button.


IMiS/wScan - Installation

Program Maintenance

Modify, repair, or remove the program.

○ Modify

Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

● Repair

Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

○ Remove

Remove IMiS/wScan from your computer.

Image 44: Selecting installation repairs

Installation repairs are executed in the next few steps. The procedure finishes with the installation of all required application features and does not require administrator intervention.

For more information see chapter 4.1.1 Installation with the Wizard.

4.5 Upgrade

When a new IMiS®/Scan version is issued, the new version must be installed on each workstation. This procedure is carried out with the Installation Wizard and matches the procedure for product installation.

During the upgrade procedure the previous version of the product is automatically uninstalled. All user settings are preserved. This is followed by the procedure of installing the new version. For more information see chapter 4.1.1 Installation with the Wizard.

5 TECHNICAL DOCUMENTATION

This documentation has been prepared for developers with knowledge of the JavaScript programming language and who are familiar with the concepts of object-oriented programming. It is divided into imis.scan.js and imis.scan.ui.js, and examples of using both libraries.

5.1 imis.scan.js

The **imis.scan.js** JavaScript library manages data exchange with IMiS®/Capture Service.
This library is built on the ECMAScript 6 standard.

This library enables different functionalities:

- Adding, reading, modifying, deleting profiles.
- Adding, reading, executing, stopping jobs.
- Reading documents.
- Reading document pages.
- Reading barcodes on page.
- Reading modules.

5.1.1 imis.scan.Scan

This object represents methods for exchanging data with the IMiS®/Capture Service server.
It enables reading, creating, updating and deleting profiles, and reading and creating jobs.
It enables the detection of creating and deleting profiles and of creating jobs.
Before starting the library, the developer must obtain a security key to access the IMiS®/Capture Service. For obtaining a security key see chapter 4.3.2 Security settings.

Constructor

imis.scan.Scan(options)	Creates a new object for data exchange and establishes a connection to the IMiS®/Capture Service server. Options object:		
	url	string	Address to IMiS®/Capture Service, the default value is the address of the host or computer (optional).
	apiKey	string	Key to access the IMiS®/Capture Service.
	onConnect	callback	Callback on successful connection to IMiS®/Capture Service (optional). callback: function()
	onConnectError	callback	Callback on unsuccessful connection to IMiS®/Capture Service (optional). callback: function(error: string)
	onError	callback	Callback on error (optional). callback: function(error: string)

Methods

connect()	Establishes a connection to the IMiS®/Capture Service server.
onCreateProfile(callback)	Callback on creating a new profile. Parameters: - callback: function(profile: imis.scan.Profile)
onDeleteProfile(callback)	Callback on deleting a profile. Parameters: - callback: function(id: String)
onCreateJob(callback)	Callback on creating a job. Parameters: - callback: function(job: imis.scan.Job)

Methods (cont.)

onError(callback)	Callback on error. Parameters: - callback: function(error: string)			
getProfile(options)	Returns a profile. Options object:	id	string	Unique profile identifier
		success	callback	Callback on successful reading of a profile. callback: function(profile: imis.scan.Profile)
		error	callback	Callback on unsuccessful reading of a profile. callback: function(error: string)
getProfiles(options)	Returns a collection of profiles. Options object:	success	callback	Callback on successful reading of a collection of profiles. callback: function(profiles: imis.scan.Profile[])
		error	callback	Callback on unsuccessful reading of a collection of profiles. callback: function(error: string)
createProfile(options)	Creates a profile. Options object:	profile	imis.scan.Profile	New profile.
		success	callback	Callback on successful creation of a profile. callback: function(profile: imis.scan.Profile)
		error	callback	Callback on unsuccessful creation of a profile. callback: function(error: string)

Methods (cont.)

updateProfile(options)	Updates a profile. Options object:	profile	imis.scan.Profile	Profile.
		success	callback	Callback on successful saving of a profile. callback: function(profile: imis.scan.Profile)
		error	callback	Callback on unsuccessful saving of a profile. callback: function(error: string)
deleteProfile(options)	Deletes a profile. Options object:	profile	imis.scan.Profile	Profile.
		success	callback	Callback on successful deletion of a profile. callback: function()
		error	callback	Callback on unsuccessful deletion of a profile. callback: function(error: string)
getJob(options)	Returns a job. Options object:	id	string	Unique job identifier.
		success	callback	Callback on successful reading of a job. callback: function(job: imis.scan.Job)
		error	callback	Callback on unsuccessful reading of a job. callback: function(error: string)
getJobs(options)	Returns jobs. Options object:	success	callback	Callback on successful reading of jobs. callback: function(job: imis.scan.Job[])
		error	callback	Callback on unsuccessful reading of jobs. callback: function(error: string)

Methods (cont.)

createJob(options)	Creates a job; after successful creation destroys the last job created. Options object:

profile	string or imis.scan.Profile	Unique profile identifier or profile.
success	callback	Callback on successful creation of a job. callback: function(job: imis.scan.Job)
error	callback	Callback on unsuccessful creation of a job. callback: function(error: string)

getModules(options)	Returns a collection of modules. Options object:

success	callback	Callback on successful reading of a collection of modules. callback: function(job: imis.scan.Module[])
error	callback	Callback on unsuccessful reading of a collection of modules. callback: function(error: string)

5.1.2 imis.scan.Profile

This object represents a profile. It enables the detection of profile modification.

Constructor

imis.scan.Profile()	Creates a new profile.

Methods

setModule(module: imis.scan.Module)	Sets the module in the profile.
setModules(modules: imis.scan.Module [])	Sets a collection of modules in the profile. A module is removed if the remove property is set to *true*, if not, it is modified or added.
addModule(module: imis.scan.Module)	Adds a new module to the profile.
removeModule(module: imis.scan.Module)	Removes a module from the profile.

Methods (cont.)

onChange(callback)	Callback on profile modification. Parameters: - callback: function(profile: imis.scan.Profile)
clone()	Returns a copy.
equals(profile: imis.scan.Profile)	Returns *true* if the profiles are equal, if not, it returns *false*.

Properties

id	string	Returns a unique profile identifier.
name	string	Returns or sets the profile name.
disabled	boolean	Returns whether the profile has been disabled.
disabledMessage	string	Returns the reason for the disabled profile.
source	string	Returns or sets the module identifier at the source.
target	string	Returns or sets the module identifier at the target.
scannerSource	imis.scan.ScannerModule	Returns the scanner module.
folderTarget	imis.scan.FolderTargetModule	Returns the target module in charge of saving.
barcodeExtractor	imis.scan.BarcodeExtractorModule	Returns the module that recognizes barcodes.
changed	Boolean	Returns whether an object has been changed.
modules	imis.scan.Module[]	Returns a collection of modules.

5.1.3 imis.scan.Job

This object represents a job, enables startup, canceling, detection of modifications to properties and detection of document creation.

Constructor

imis.scan.Job(options)	Creates a new object.

Methods

start(options)	Starts a job. Options object: success | callback | Callback on successfully starting a job. callback: function(job: imis.scan.Job) error | callback | Callback on unsuccessfully starting a job. callback: function(error: string)
cancel(options)	Cancels a job. Options object: success | callback | Callback on successfully canceling a job. callback: function(job: imis.scan.Job) error | callback | Callback on unsuccessfully canceling a job. callback: function(error: string)
onChange(callback)	Callback on changing a job. Parameters: - callback: function(job: imis.scan.Job)
onCreateDocument(callback)	Callback on creating a document. Parameters: - callback: function(document: imis.scan.Document)
getJob(options)	Returns a job. Options object: success | Callback | Callback on successful reading of a job. callback: function(job: imis.scan.Job) error | callback | Callback on unsuccessful reading of a a job.

				callback: function(error: string)
getDocuments(options)	Returns a collection of documents. Options object:			
		success	callback	Callback on successful reading of documents. callback: function(documents: imis.scan.Document[])
		error	callback	Callback on unsuccessful reading of documents. callback: function(error: string)
getDocument(options)	Returns a document. Options object:			
		success	callback	Callback on successful reading of a document. callback: function(document: imis.scan.Document)
		error	callback	Callback on unsuccessful reading of a document. callback: function(error: string)
getNextDocument(document: imis.scan.Document)	Returns the next document imis.scan.Document if it exists.			
getPrevDocument(document: imis.scan.Document)	Returns the previous document imis.scan.Document if it exists.			
onError(callback)	Callback on job error. Parameters: - callback: function(error: string)			

Methods (cont.)

destroy()	Destroys a job.
download(callback, errorCallback)	Returns the URL to transfer all documents in a task. Parameters: - callback: function(uri: string) - errorCallback: function()
isCompleted()	Returns whether the task has been completed.
isCancelled()	Returns whether the task has been cancelled.
isCreated()	Returns whether the task has been created.
isInProgress()	Returns whether the task is in progress.
isPending()	Returns whether the task is pending.
isError()	Returns whether an error occurred.

Properties

id	string	Returns a unique job identifier.
index	number	Returns a job index.
name	string	Returns a job name.
status	number	Returns job status. Set of values: - -2: Job has been canceled. - -1: An error has occurred. - 0: Job has been finished. - 1: Job has been created. - 2: Job has been queued. - 3: Job is being executed.
error	string	Returns a job error message.
created	string	Returns the date and time of job creation. Format 2017-10-02T09:58:15.9225533+02:00.
documentCount	number	Returns the number of documents in a job.
pageCount	number	Returns the number of pages in all documents in a job.
fileName	string	Returns the name of the task file.

5.1.4 imis.scan.Document

This object represents a document, enables the detection of creating pages within a document, changes to document properties, and reading of pages.

Methods

getFirstPage()	Returns the first page imis.scan.Page in a document.
getLastPage()	Returns the last page imis.scan.Page in a document.
getNextPage(page: imis.scan.Page)	Returns the next page imis.scan.Page in a document.
getPrevPage(page: imis.scan.Page)	Returns the previous page imis.scan.Page in a document.

Methods (cont.)

onChange(callback)	Callback on changing a document. Returns the same document with changed properties. Parameters: - callback: function(document: imis.scan.Document)
onCreatePage(callback)	Callback on creating a page within a document. Parameters: - callback: function(page: imis.scan.Page)
onError(callback)	Callback on document error. Parameters: - callback: function(error: string)
destroy()	Destroys a document.
download(callback, errorCallback)	Obtains a link for document transfer. Parameters: - callback: function(uri: string) - errorCallback: function()

Properties

id	String	Returns a unique document identifier.
name	String	Returns a document name.

Properties (cont.)

index	Number	Returns a document index.
mime	String	Returns the type of document content.
pageCount	Number	Returns the number of pages in a document.
created	String	Returns the date and time of creation. Format: 2017-10-02T09:56:26.4618227+02:00
length	Number	Returns the size of the document in bytes.
fileName	String	Returns the name of the task file.

5.1.5 imis.scan.Page

This object represents a page, enables the detection of changes to properties, reading a page preview, and reading a page in basic size.

Methods

getThumbnailUri(options)	Obtains the link to the page preview URI in the image/png format. Options object:

height	Number	Height of page preview.
width	Number	Width of page preview.
success	Callback	Call when connection was successful. Parameters: - callback: function(uri: string)
error	Callback	Call when connection failed. Parameters: - callback: function()

getImage(callback, errorCallback)	Obtains the link to the page URI in the image/png format. Parameters: - callback: function(uri: string) - errorCallback: function()
onChange(callback)	Callback on changing a page. Returns the page with changed properties. Parameters: - callback: function(page: imis.scan.Page)

Properties

id	string	Returns a unique page identifier.
index	number	Returns a page index.
width	number	Returns the page width.
height	number	Returns the page height.
xresolution	number	Returns the horizontal page resolution in DPI.
yresolution	number	Returns the vertical page resolution in DPI.
barcodes	imis.scan.Barcode[]	Returns the collection of barcodes on the page.
colorFormat	imis.scan.ColorFormat	Returns the page color format.

5.1.6 imis.scan.Barcode

This object represents the properties of the barcode recognized on a page.

Properties

height	number	Returns the barcode height.
width	number	Returns the barcode width.
text	string	Returns the recognized barcode content.
point1	number	Returns the top left point of the barcode.
point2	number	Returns the top right point of the barcode.
point3	number	Returns the bottom right point of the barcode.
point4	number	Returns the bottom left point of the barcode.
posX	number	Returns the horizontal offset of the barcode on the page.
posY	number	Returns the vertical offset of the barcode on the page.
type	string	Returns the barcode type.

5.1.7 imis.scan.Module

This object represents the basis of the module, from which different modules have been derived.

Methods

clone()	Returns a copy.

Properties

id	string	Returns a unique module identifier.
sendTo	string[]	Returns or sets a collection of module identifiers, to which the module sends data.
type	string	Returns the module type. Set of values: - Scanner_source - Barcode_extractor - Folder_target.
remove	boolean	Returns or sets the value that determines whether the module will be removed; applies only when calling the setModules method on imis.scan.Profile.

5.1.8 imis.scan.ScannerModule

This object represents the scanner module, which enables the reading and changing of scanner settings.

This class has been derived from imis.scan.Module.

Properties

driverName	string	Returns or sets the driver name.
scannerModel	string	Returns the scanner model.
scannerValues	imis.scan.ScannerValue[]	Returns a collection of all connected scanners.
paperSize	string	Returns or sets the paper size.
paperSizes	string[]	Returns the collection of paper sizes for the selected driver (driverName).
resolution	number	Returns or sets the scanning resolution.
resolutions	number[]	Returns a collection of scanning resolutions for the selected driver (driverName).
colorFormat	imis.scan.ColorFormat	Returns or sets the scanning color.
colorFormats	imis.scan.ColorFormat[]	Returns a collection of scanning colors for the selected driver (driverName).
duplex	boolean	Returns or sets whether duplex scanning is enabled.

5.1.9 imis.scan.FolderTargetModule

This object represents the target module, which enables the setting of properties of saving files to the file system, with the option of setting the directory, file format, color and compression.

This class has been derived from imis.scan.Module.

Properties

folder	string	Returns or sets the path to the directory.
fileRoot	string	Returns or sets the file name.
fileFormat	string	Returns or sets the file format.
fileFormats	string[]	Returns a set of file formats. Set of values: - BMP - GIF - TIFF - JPEG - PCX - PDF/A - PNG.
colorFormat	imis.scan.ColorFormat	Returns or sets the color format.
colorFormats	imis.scan.ColorFormat[]	Returns a collection of color formats; this collection is connected to the file format (fileFormat).
compression	string	Returns or sets the compression.
compressions	string[]	Returns a collection of compressions; this collection is connected to the color format (colorFormat) and file format.

5.1.10 imis.scan.BarcodeExtractorModule

This object represents the module which enables the detection of barcodes on an individual page.

This class has been derived from imis.scan.Module.

Properties

types	string[]	Returns or sets a collection of barcode types for recognition.
typesValues	string[]	Returns a collection of barcode types. Set of values: - addon2 - addon5 - australianpost - aztec - bcdmatrix - codabar - code25_datalogic - code25_iata - code25_industrial - code25_interleaved - code25_invert - code25_matrix - code32 - code39 - code93 - datamatrix - ean13 - ean8 - intelligentmail - pdf417 - postnet - qrcode - royalpost - type128 - ucc128 - upc_a - upc_e.
orientation	string	Returns or sets the option of barcode orientation.
orientationValues	string[]	Returns a collection of barcode orientations. Set of values: - horizontal: Detection of horizontal barcodes. - vertical: Detection of vertical barcodes. - both: Detection of horizontal or vertical barcodes. - horizontalverzicaldiagonal: Detection of horizontal, vertical or 45° barcodes.

Properties (cont.)

mode	string	Returns or sets the barcode detection mode.
modeValues	string[]	Returns a collection of barcode detection modes. Set of values: - normal: Normal mode, faster than enhanced. - enhanced: Improved mode, enables better detection; detection is slower.

5.1.11 imis.scan.ScannerValue

This object represents scanner properties.

Properties

driverName	string	Returns the driver name.
scannerModel	string	Returns the scanner model.
colorFormats	imis.scan.ColorFormat[]	Returns a collection of colors.
paperSizes	string[]	Returns a collection of paper sizes.
resolutions	number[]	Returns a collection of resolutions.

5.1.12 imis.scan.ColorFormat

This object represents the properties of the color format.

Properties

colorMode	string	Returns the image color type. Set of values: - blackwhite: Black-and-white image; - grayscale: A grayscale image; - color: A color image.
colorDepth	number	Returns the image color depth.
photometric	string	Returns the mode of reading image data.
compressions	string[]	Returns a collection of compressions; this collection exists only when reading the properties of colorFormats on imis.scan.FolderTargetModule.

5.2 imis.scan.ui.js

This library enables easy use of components, which can be used for displaying scanning.

The displaying of components is managed by the main component imis.scan.ui.Scan, where we specify all of the components we will be using.

The imis.scan.js library is required for it to work.

5.2.1 imis.scan.ui.Scan

This object represents the main component, which manages the displaying of various components. When creating this component, the components are initialized.

Constructor

imis.scan.ui.Scan(options: UIScanOptions)	Creates a new object and components are initialized.

Methods

show()	Establishes a connection to the IMiS®/Capture Service server and data is loaded into the components.

Properties

job	imis.scan.Job	Current job.

5.2.1.1 UIScanOptions

This object represents the imis.scan.ui.Scan settings options.

Properties

url	string	Address to IMiS®/Capture Service. The default value is the address of the host or computer (optional).
notifications	boolean	Displaying notifications in a browser; the default value is true (optional).
apiKey	string	Key to access the IMiS®/Capture Service.
thumbnails	imis.scan.ui.Thumbnails	Displaying documents and pages (optional).
settings	imis.scan.ui.Settings	The settings of all profiles (optional).

Properties (cont.)

imageView	imis.scan.ui.ImageView	Displaying a selected page (optional).
images	imis.scan.ui.ImageScroll	Displaying all pages (optional).
status	imis.scan.ui.Status	Displaying status (optional).
imageDetails	imis.scan.ui.ImageDetails	Details of a selected page (optional).
progress	imis.scan.ui.Progress	Displaying job status (optional).
buttons	UIScanButtonsOptions	Buttons settings.
useLocalStorage	boolean	Specifies the use of saving settings (currently selected profile) to the browser; if the value is set to false, the saved settings are deleted; the default value is true (optional).

5.2.1.2 UIScanButtonsOptions

This object represents the settings options for buttons.

Properties

scan	imis.scan.ui.Button	The button for starting a scan. The default text value is Scan.
continue	imis.scan.ui.Button	The button for continuing the scan (optional). The default text value is Continue.
cancel	imis.scan.ui.Button	The button for canceling the scan (optional). The default text value is Cancel.
download	imis.scan.ui.Button	The button for downloading all scanned documents (optional). The default text value is Save.
color	imis.scan.ui.ColorDropdownButton	The list of possible colors if a scanner is available; changes are saved only temporarily for each job started (optional).
profiles	imis.scan.ui.ProfilesButton	A collection of profiles and the option of temporarily editing the selected profile; changes are saved only for each job started (optional).

5.2.2 imis.scan.ui.Button

This object represents the button component, which represents the basic component used to control the start, continuation, cancellation and download of a job.

Image 45: Button component

Constructor

imis.scan.ui.Button(options)	Creates a new button. Options object:		
	id	string	Unique identifier of an element in an HTML document as the id attribute.
	text	string	Button text (optional).
	tooltip	string	Contents of the pop-up notification below the button (optional).
	darkMode	boolean	A dark display mode. Default value: *false* (optional).
	color	string	Color of button text (optional).
	backgroundCo lor	string	Color of button background (optional).
	fontSize	string	Size of button text (optional).
	width	string	Minimum button width (optional).
	height	string	Button height (optional).
	onClick	callback	Call when clicking the button (optional). Parameters: - callback: function()

Methods

disable()	Disable button.
enable()	Enable button.
showProgress()	Shows progress within the button.
hideProgress()	Hides progress within the button.

Properties

app	imis.scan.ui.Scan	Returns the main component.

5.2.3 imis.scan.ui.ColorDropdownButton

This object represents the component for selecting the color of scanning. This value only affects the job which will be started by pressing the scan button. This component will display values if a scanner module exists in the selected profile and if it has color selection values.

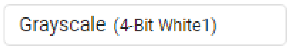


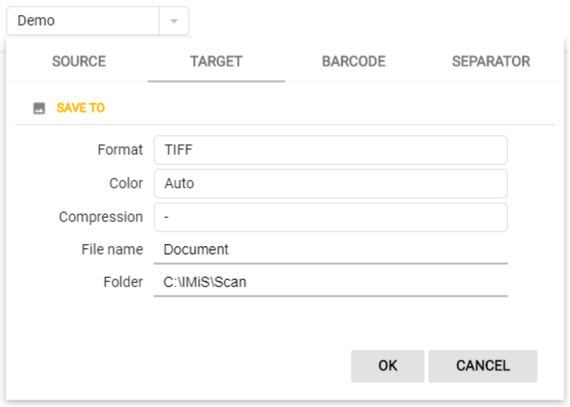
Image 46: Component for selecting the color of scanning

Constructor

imis.scan.ui.ColorDropdownButton(options)	Creates a new dropdown menu for selecting the color of scanning. Options object:		
	id	string	Unique identifier of an element in an HTML document as the id attribute.

5.2.4 imis.scan.ui.ProfilesButton

This object represents the component for selecting the profile and changing the settings of the scanning profile. Changes to the profile will be applied only at the start of each new job.
The selected profile will be saved to the browser (Local Storage), if saving has been enabled.



Format: TIFF
Color: Auto
Compression: -
File name: Document
Folder: C:\IMiS\Scan

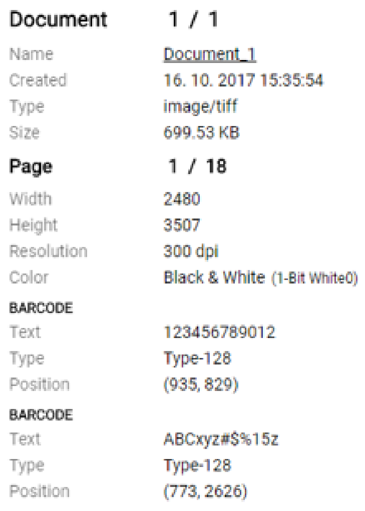
Image 47: Component for selecting the profile and changing the settings of the scanning profile

Constructor

imis.scan.ui.ProfilesButton(options)	Creates a new dropdown menu for selecting the profile with the option of changing the settings for a new job. Options object: id | string | Unique identifier of an element in an HTML document as the id attribute.

5.2.5 imis.scan.ui.ImageDetails

This object represents the component for displaying information about the currently selected page. For it to work, the component imis.scan.ui.ImageView or imis.scan.ui.ImageScroll in imis.scan.ui.Scan must be used.



Document	**1 / 1**
Name	Document_1
Created	16. 10. 2017 15:35:54
Type	image/tiff
Size	699.53 KB
Page	**1 / 18**
Width	2480
Height	3507
Resolution	300 dpi
Color	Black & White (1-Bit White0)
BARCODE	
Text	123456789012
Type	Type-128
Position	(935, 829)
BARCODE	
Text	ABCxyz#$%15z
Type	Type-128
Position	(773, 2626)

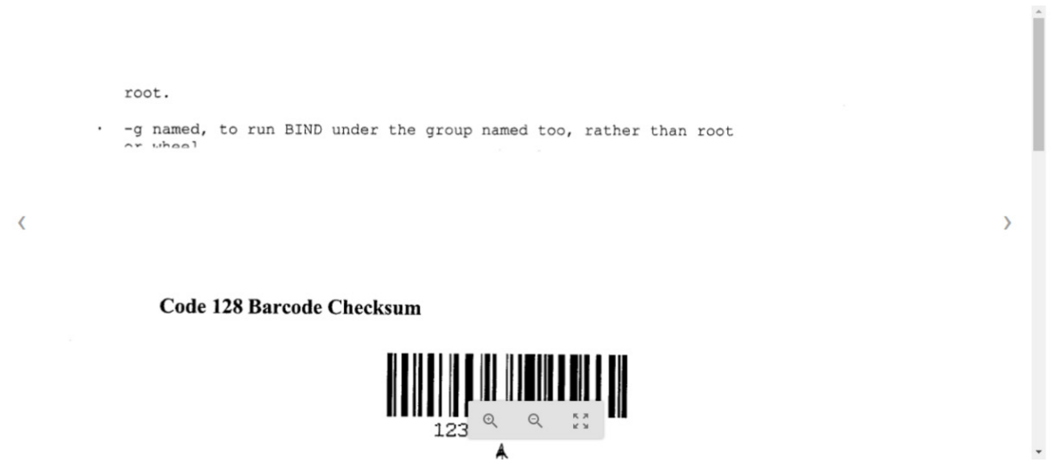
Image 48: Component for displaying information about the currently selected page

Constructor

imis.scan.ui.ImageDetails(options)	Creates a component for displaying details of the selected page. Options object:		
	id	string	Unique identifier of an element in an HTML document as the id attribute.
	closed	boolean	Specifies whether a component is hidden. Default value: *false* (optional).
	close	boolean	Specifies whether the close button is displayed. Default value: *true* (optional).
	darkMode	boolean	A dark display mode. Default value: *false* (optional).
	onClose	callback	Callback on clicking on the close button (optional).
	background	string	Background color (optional).
	color	string	Text color (optional).

5.2.6 imis.scan.ui.ImageView

This object represents the component for displaying the currently selected page. This component enables changing the size (zooming in/zooming out or showing the whole page) and moving to the next or previous page.



```
root.

• -g named, to run BIND under the group named too, rather than root
```

Code 128 Barcode Checksum

123

Image 49: Component for displaying the currently selected page

Constructor

imis.scan.ui.ImageView(options)	Creates a component for displaying the selected page. Options object: id | string | Unique identifier of an element in an HTML document as the id attribute. onPropertiesSelected | callback | Callback on selecting properties in the context menu (optional). callback: function() background | string | Background color (optional).

5.2.7 imis.scan.ui.ImageScroll

This object represents the component for displaying a collection of pages. This component enables changing the size (zooming in/zooming out or showing the whole page) and moving to the next or previous page.

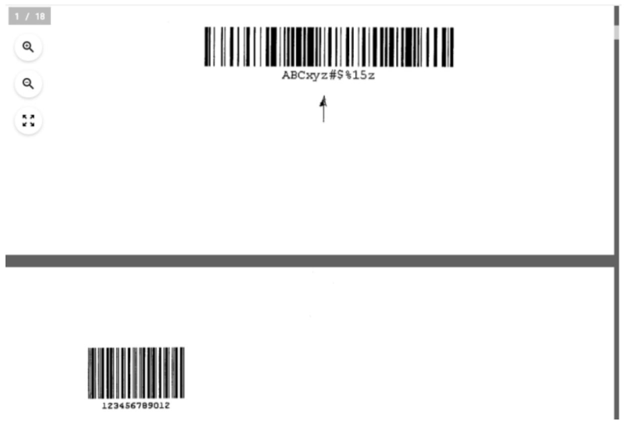


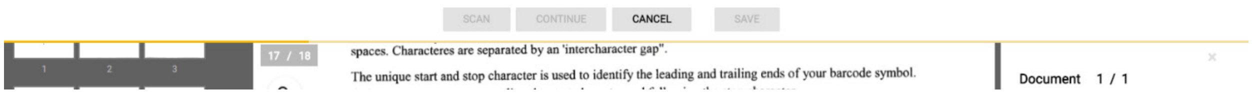
Image 50: Component for displaying a collection of pages

Constructor

imis.scan.ui.ImageScroll(options)	Creates a component for displaying a collection of pages. Options object:		
	id	string	Unique identifier of an element in an HTML document as the id attribute.
	focusNewPage	boolean	When adding a new page, it is shown in focus. Default value: *true* (optional).
	pageIndex	boolean	Displaying the position of the current page. Default value: *true* (optional).
	controls	boolean	Displaying the controls for zooming in, zooming out, and adjusting the image to fit the screen. Default value: *true* (optional).
	darkMode	boolean	A darker display mode. Default value: *false* (optional).
	contextMenu.enabled	boolean	Specifies whether a context menu for an individual page (optional) has been enabled.
	contextMenu.onPropertiesSelected	callback	Callback on selecting properties in the context menu (optional). callback: function()
	background	string	Background color (optional).

5.2.8 imis.scan.ui.Progress

This object represents the component for showing the progress of the current job.



spaces. Characteres are separated by an 'intercharacter gap".

The unique start and stop character is used to identify the leading and trailing ends of your barcode symbol.

Image 51: Component for showing the current job

Constructor

imis.scan.ui.Progress(options)	Creates a component for showing the progress of the current job. Options object:		
	id	string	Unique identifier of an element in an HTML document as the id attribute.
	darkMode	boolean	A dark display mode. Default value: *false* (optional).
	color	string	Component color while executing the job (optional).

5.2.9 imis.scan.ui.Status

This object represents the component for displaying the status, which specifies whether a server connection has been established.

Image 52: Component for displaying the status

Constructor

imis.scan.ui.Status(options)	Creates a component for displaying the status. Options object:		
	id	string	Unique identifier of an element in an HTML document.

5.2.10 imis.scan.ui.Thumbnails

This object represents the component for showing documents and pages. This component enables adjusting the size of individual page previews, orientation of a collection of documents, or the gallery mode, which shows the page and details in a dialog.

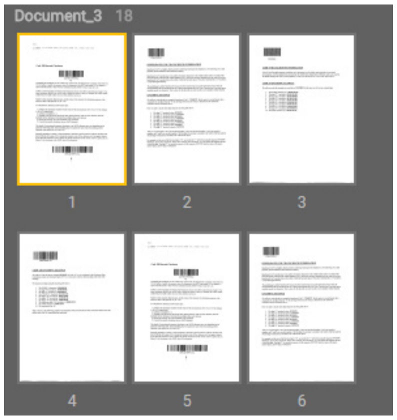


Image 53: Component for showing documents

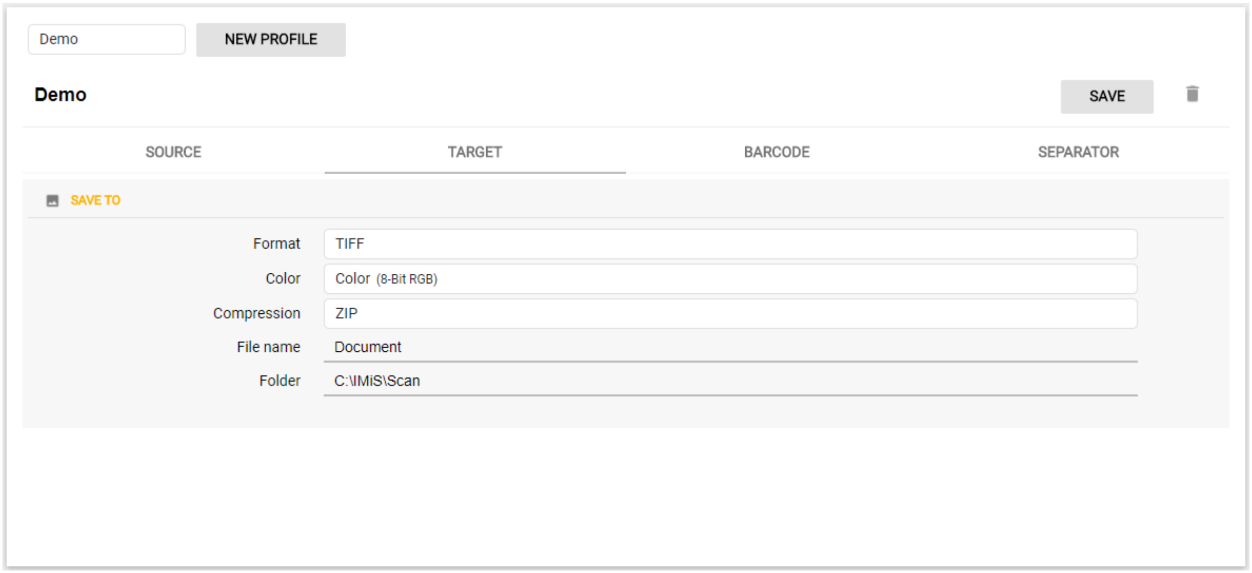
Constructor

imis.scan.ui.Thumbnails (options)	Creates a component for showing the page as a preview. Options object:		
	id	string	Unique identifier of an element in an HTML document.
	gallery	boolean	The gallery mode. Enables double-clicking on an individual page, which opens a dialog with the zoomed-in page and details. Default value: *false* (optional).
	focusNewPage	boolean	When adding a new page, it is shown in focus. Default value: *true* (optional).
	darkMode	boolean	A darker display mode. Default value: *false* (optional).
	orientation	string	Orientation of a collection of documents. Set of values: - horizontal - vertical. Default value: horizontal (optional).
	thumbnail.width	number	Page width (optional).
	thumbnail.height	number	Page height (optional). Default value: 150.
	thumbnail.title	boolean	Specifies whether the page title is shown. Default value: *true* (optional).
	thumbnail.titleColor	string	Color of the page title (optional).

	contextMenu.enable d	boolean	Specifies whether a context menu for an individual page (optional) has been enabled.
	contextMenu. onPropertiesSelecte d	callback	Callback on selecting properties in the context menu (optional). callback: function()
	color	string	Color of the document title (optional).
	backgroundColor	string	Background color (optional).

5.2.11 imis.scan.ui.Settings

This object represents the component for setting profiles. This component enables showing, adding, changing or deleting profiles.



Format	TIFF
Color	Color (8-Bit RGB)
Compression	ZIP
File name	Document
Folder	C:\IMiS\Scan

Image 54: Component for setting profiles

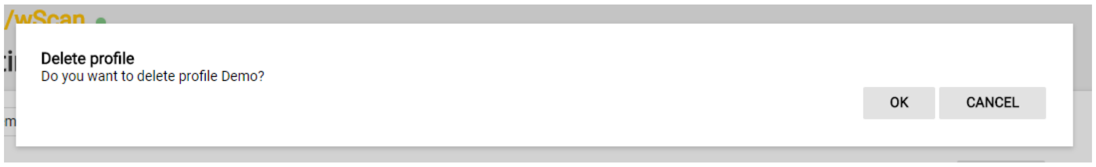
Constructor

imis.scan.ui.Settings(options)	Creates and displays a new component for showing the settings. Options object:

id	string	Unique identifier of an element in an HTML document.

5.2.12 imis.scan.ui.AlertDialog

This object represents the component for displaying a dialog. This component enables setting the title and text, and detecting whether a user has confirmed or canceled a dialog.



Delete profile
Do you want to delete profile Demo?

Image 55: Component for displaying a dialog

Constructor

imis.scan.ui.AlertDialog(options)	Creates and displays a new dialog. Options object:

title	string	Dialog title.
text	string	Dialog content.
ok	callback	Callback on pressing the ok button. callback: function()
cancel	callback	Callback on pressing the cancel button. callback: function()

5.3 Examples of use imis.scan.js

These examples show the use of the “imis.scan.js” library. Developers can easily learn to use the library with the help of these prepared examples. Examples of reading profiles, modifying a profile, starting a job and deleting a job are shown. These examples can be accessed from the start page if they were turned on during installation and are executable.

5.3.1 Reading profiles

An example of reading profiles shows the basic use of the library. If reading is successful, a collection of profiles will appear in the feature with the ‘profiles’ identifier; if an error occurs, it will be visible in the feature with the ‘error’ identifier. When the page has finished loading, a scan object will be created, which will be used to read all the profiles. If they were read successfully, they will appear on a list, if not, an error will appear.

```
<!DOCTYPE html>
<html>
<head>
  <title>imis.scan.js</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="sample.css" />
</head>
<body class="sample">
  <h1>Sample</h1>
  <p>This example demonstrates reading scan profiles.</p>

  <div>Profiles:</div>
  <ol id="profiles"></ol>
  <div id="error"></div>

  <script src="../imis.scan.js"></script>
  <script>
    window.addEventListener('load', function () {
      try {
        // Profiles ordered list
        var ol = document.getElementById("profiles");

        // Create a scan object
        var scan = new imis.scan.Scan(
         apiKey = API_KEY);

        // Read profiles
        scan.getProfiles({
          success: function (profiles) {
            for (var i = 0; i < profiles.length; i++) {
              // Add profile to ordered list
              var li = document.createElement("li");
              li.innerHTML = profiles[i].name;
              ol.appendChild(li);
            }
          },
          error: function (error) {
            // Show error
            document.getElementById("error").innerHTML = error;
          }
        });
      } catch (e) {
        // Show error
        document.getElementById("error").innerHTML = e;
      }
    });
  </script>
</body>
</html>
```

5.3.2 Changing a profile

An example of changing the profile name shows the basis for changing profile properties.

A collection of all profiles is loaded to the 'select' feature with the 'profiles' identifier, where you select the current profile to be changed. Clicking on the 'button' feature with the 'btn-update' identifier initiates a profile update and updates the collection of profiles. When the page has finished loading, a scan object will be created, which will be used to read all the profiles.

If they were read successfully, they will appear on a list and the name of the selected profile will be loaded to the input field, if not, an error will appear. When changing the profile in the drop-down menu, the input field will be updated with the profile name. When pressing the Update button, the new profile name will be saved to the profile and this change will be saved to the server; if saved successfully, the list of profiles will be updated, if not, an error will appear.

```
<!DOCTYPE html>
<html>
<head>
  <title>imis.scan.js</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="sample.css" />
</head>
<body class="sample">
  <h1>Sample</h1>
  <p>This example demonstrates updating profile name.</p>

  <h2>Update profile</h2>
  Select profile <select id="profiles"></select>

  <h3>Edit</h3>
  Profile name <input id="profile-name" type="text" placeholder="Profile name" />
  <button id="btn-update">Update</button>
  <div id="error"></div>

  <script src="../imis.scan.js"></script>
  <script>
    window.addEventListener('load', function () {
      var profilesList = [], // Profiles list
          selectedProfile = null, // Selected profile
          profilesSelectUI = document.getElementById("profiles"), // Profiles drop-
down list
```

```
          profileNameUI = document.getElementById("profile-name"); // Selected profile
name input text

      try {
        // Create a scan object
        var scan = new imis.scan.Scan(
         apiKey = API_KEY);

        // Load profiles to drop-down list
        var load = function () {
          scan.getProfiles({
            success: function (profiles) {
              profilesList = profiles;
              // Clear options
              profilesSelectUI.innerHTML = "";

              for (var i = 0; i < profiles.length; i++) {
                // Add profile option
                const profile = profiles[i];
                var option = document.createElement("option");
                option.value = profile.id;
                option.text = profile.name;
                if (null === selectedProfile)
                  selectedProfile = profile;
                option.selected = profile.equals(selectedProfile)
                profilesSelectUI.add(option);
              }

              // Update selected profile name text input
              if (null !== selectedProfile)
                profileNameUI.value = selectedProfile.name;
            },
            error: function (error) {
              // Show error
              document.getElementById("error").innerHTML = error;
            }
          });
        };
        // Call load
        load();

        // Selected profile change listener
        profilesSelectUI.addEventListener("change", function () {
          // Update selected profile
          selectedProfile = null;
          var selectedValue =
profilesSelectUI.options[profilesSelectUI.selectedIndex].value;
          for (var i = 0; i < profilesList.length; i++) {
            if (profilesList[i].id === selectedValue) {
              selectedProfile = profilesList[i];
              break;
            }
          }

          // Update selected profile name text input
          if (null !== selectedProfile)
            profileNameUI.value = selectedProfile.name;
        });

        // Update button click listener
        document.getElementById("btn-update").addEventListener("click", function () {
          if (null == selectedProfile || null === profileNameUI.value || "" ===
profileNameUI.value ||
            selectedProfile.name === profileNameUI.value)
            return;

          // Update profile name
          selectedProfile.name = profileNameUI.value;

          // Save profile
          scan.updateProfile({
            profile: selectedProfile,
            success: function (profile) {
              // Load profiles
              load();
            },
            error: function (error) {
              // Show error
              document.getElementById("error").innerHTML = error;
            }
          });
        });
      } catch (e) {
        // Show error
        document.getElementById("error").innerHTML = e;
      }
    });
  </script>
</body>
</html>
```

5.3.3 Starting a job

An example of starting a job; shows job management and displays the scanning result.

A collection of all profiles is loaded to the 'select' feature with the 'profiles' identifier, where you select the current profile to create and start a job. Clicking on the 'button' feature with the 'btn-start' identifier initiates the creation and start of a job.

Status is shown in the feature with the 'job-progress' identifier. The entire job can be downloaded by clicking on the feature with the 'job-download' identifier, after the job has been finished.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>imis.scan.js</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="sample.css" />
</head>
<body class="sample">
  <h1>Sample</h1>
  <p>This example demonstrates starting job and displaying documents and pages.</p>

  <h2>Start Job</h2>
  <select id="profiles"></select>
  <button id="btn-start">Start</button>
  <br /><br />

  <div>Status: <span id="job-progress">None</span></div>
  <div id="error"></div>
  <a id="job-download" href="#">Download</a>
  <br /><br />

  <div><b>Documents</b></div>
  <div id="job">None</div>

  <script src="../imis.scan.js"></script>
  <script>
    window.addEventListener('load', function () {
      var profilesList = [], // Profiles list
          profilesSelectUI = document.getElementById("profiles"), // Profiles drop-
down list
          jobUI = document.getElementById("job"); // Job documents

      try {
        // Create a scan object
        var scan = new imis.scan.Scan(
         apiKey = API_KEY);

        // Get profiles
        scan.getProfiles({
          success: function (profiles) {
            profilesList = profiles;
            // Clear options
            profilesSelectUI.innerHTML = "";

            for (var i = 0; i < profiles.length; i++) {
              // Add profile option
              const profile = profiles[i];
              var option = document.createElement("option");
              option.value = profile.id;
              option.text = profile.name;
              profilesSelectUI.add(option);
            }
          },
          error: function (error) {
            console.error("getProfiles: " + error);
          }
        });

        // Start job button click listener
        document.getElementById("btn-start").addEventListener("click", function () {
          // Clear download link
          document.getElementById("job-download").setAttribute("href", "#");

          // Find selected profile
          var selectedValue =
profilesSelectUI.options[profilesSelectUI.selectedIndex].value;
          var profile = null;
          for (var i = 0; i < profilesList.length; i++) {
            if (profilesList[i].id === selectedValue) {
              profile = profilesList[i];
              break;
            }
          }
          if (null == profile)
            return;

          // Create job
          scan.createJob({
            profile: profile.id,
            success: function (job) { // Job successfully created
```

```
              // Clear job documents
              jobUI.innerHTML = "";

              // Start job
              job.start({
                success: function () { // Job successfully started
                  // Update job progress
                  document.getElementById("job-progress").innerHTML =
imis.scan.JobStatus.toString[job.status];

                  // Job changed callback
                  job.onChange(function (job) {
                    // Update job progress
                    document.getElementById("job-progress").innerHTML =
imis.scan.JobStatus.toString[job.status];
                    if (imis.scan.JobStatus.COMPLETED === job.status)
                      document.getElementById("job-download").setAttribute("href",
job.download);
                  });

                  // Document created callback
                  job.onCreateDocument(function (newDocument) {
                    const documentElement = document.createElement("div");
                    documentElement.style.marginBottom = "55px";

                    // Document name
                    const documentName = document.createElement("div");
                    documentName.style.fontWeight = "bold";
                    documentName.style.fontSize = "16px";
                    documentName.innerHTML = newDocument.name + " [" +
newDocument.pageCount + "]";
                    documentElement.appendChild(documentName);
                    jobUI.appendChild(documentElement);

                    // On create page callback
                    newDocument.onCreatePage(function (page) {
                      const pageElement = document.createElement("div");
                      pageElement.style.display = "inline-block";

                      // Page image
                      const img = document.createElement("img");
                      img.setAttribute("src", page.getThumbnailUri({ height: 150 }));
                      pageElement.appendChild(img);

                      // Page index
                      const pageIndex = document.createElement("div");
                      pageIndex.innerHTML = "Page " + page.index;
                      pageElement.appendChild(pageIndex);

                      documentElement.appendChild(pageElement);
                    });

                    // Document change callback
                    newDocument.onChange(function (changedDocument) {
                      documentName.innerHTML = changedDocument.name + " [" +
changedDocument.pageCount + "]";
                    });
                  });
                },
                error: function (error) {
                  // Show error
                  document.getElementById("error").innerHTML = error;
                }
              });
            },
            error: function (error) {
              // Show error
              document.getElementById("error").innerHTML = error;
            }
          });
        });
      } catch (e) {
        // Show error
        document.getElementById("error").innerHTML = e;
      }
    });
  </script>
</body>
</html>
```

5.3.4 Deleting a profile

An example of deleting a profile shows the basic use of the library. If read successfully, the collection of profiles will appear in the drop-down menu, where you can select the profile for deletion. The profile will be deleted by clicking on the delete button and confirming the deletion dialog. This example deletes a profile saved on the server.

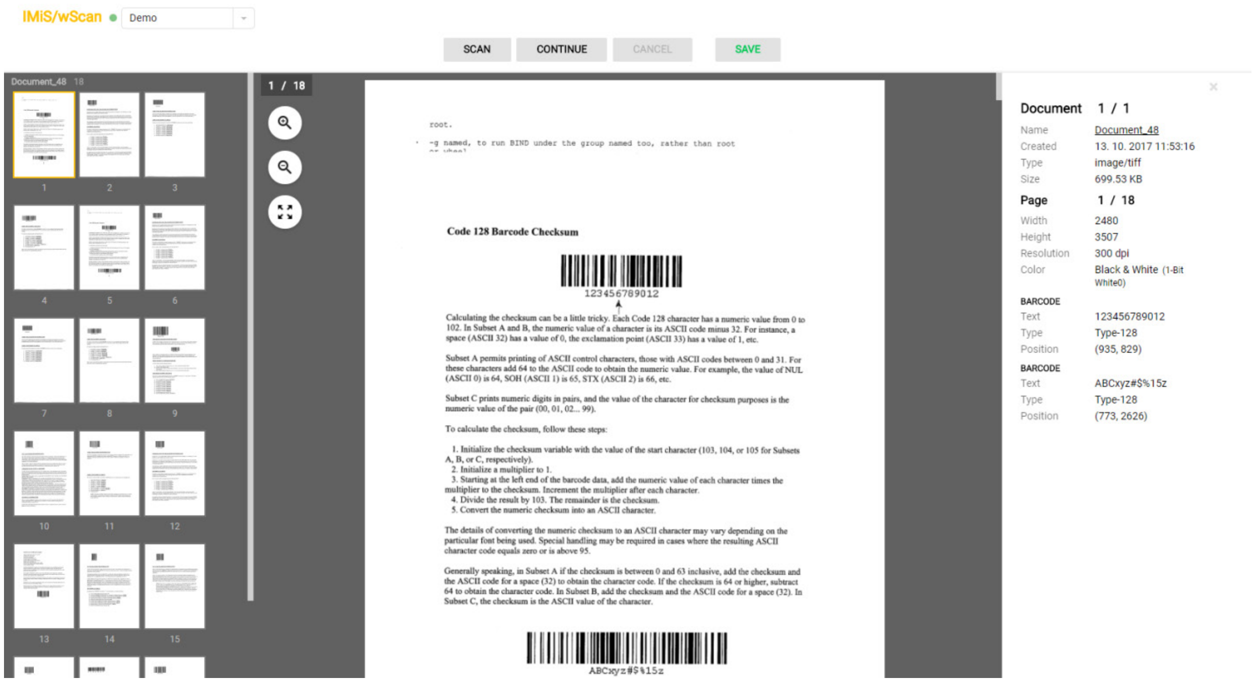
```
<!DOCTYPE html>
<html>
<head>
  <title>imis.scan.js</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="sample.css" />
</head>
<body class="sample">
  <h1>Sample</h1>
  <p>This example demonstrates deleting profile.</p>

  <h2>Delete profile</h2>
  <select id="profiles"></select>
  <button id="btn-delete">Delete</button>
  <div id="error"></div>

  <script src="../imis.scan.js"></script>
  <script>
  window.addEventListener('load', function () {
    var profilesList = [], // Profiles list
        profilesSelectUI = document.getElementById("profiles");  // Profiles drop-down
list

    try {
      // Create a scan object
      var scan = new imis.scan.Scan(
       apiKey = API_KEY);

      // Load profiles to drop-down list
      var load = function () {
        scan.getProfiles({
          success: function (profiles) {
            profilesList = profiles;
            // Clear options
            profilesSelectUI.innerHTML = "";

            for (var i = 0; i < profiles.length; i++) {
              // Add profile option
              const profile = profiles[i];
              var option = document.createElement("option");
              option.value = profile.id;
              option.text = profile.name;
              profilesSelectUI.add(option);
            }
          },
          error: function (error) {
            // Show error
            document.getElementById("error").innerHTML = error;
          }
        });
      }
      // Call load
      load();

      document.getElementById("btn-delete").addEventListener("click", function () {
        // Find selected profile
        var selectedValue =
profilesSelectUI.options[profilesSelectUI.selectedIndex].value;
        var profile = null;
        for (var i = 0; i < profilesList.length; i++) {
          if (profilesList[i].id === selectedValue) {
            profile = profilesList[i];
            break;
          }
        }
        if (null == profile)
          return;

        // Show confirmation dialog
        if (confirm("Do you want to delete profile '" + profile.name + "'?")) {
          // Delete profile
          scan.deleteProfile({
            profile: profile,
            success: function () {
              // Load profiles
              load();
            },
            error: function (error) {
              // Show error
              document.getElementById("error").innerHTML = error;
            }
          });
        }
      });
    } catch (e) {
      // Show error
      document.getElementById("error").innerHTML = e;
    }
  });
  </script>
</body>
</html>
```

5.4 Examples of use imis.scan.ui.js

These examples show the use of the "imis.scan.ui.js" library. Developers can easily learn to use the library with the help of these prepared examples. It shows examples of classic, modern, classic dark and gallery sample. These examples use different components with a specific position on the page. The developers can create a customized page layout from individual components.
These examples can be accessed from the start page if they were turned on during installation.

5.4.1 Classic sample

Classic sample is the most commonly used user interface example. Application developers choose it when they want to preserve the traditional appearance of the user interface. The thumbnails of document pages are located on the left side of the user interface. They are sorted by available space and thumbnail size (portrait, landscape). Document pages are shown in the center.
The user navigates between pages with the slider. Page details are sorted on the right and can be easily closed or reopened by selecting the menu on an individual page.



```
root.

-g named, to run BIND under the group named too, rather than root
```

Code 128 Barcode Checksum

123456789012

Calculating the checksum can be a little tricky. Each Code 128 character has a numeric value from 0 to 102. In Subset A and B, the numeric value of a character is its ASCII code minus 32. For instance, a space (ASCII 32) has a value of 0, the exclamation point (ASCII 33) has a value of 1, etc.

Subset A permits printing of ASCII control characters, those with ASCII codes between 0 and 31. For these characters add 64 to the ASCII code to obtain the numeric value. For example, the value of NUL (ASCII 0) is 64, SOH (ASCII 1) is 65, STX (ASCII 2) is 66, etc.

Subset C prints numeric digits in pairs, and the value of the character for checksum purposes is the numeric value of the pair (00, 01, 02... 99).

To calculate the checksum, follow these steps:

1. Initialize the checksum variable with the value of the start character (103, 104, or 105 for Subsets A, B, or C, respectively).
2. Initialize a multiplier to 1.
3. Starting at the left end of the barcode data, add the numeric value of each character times the multiplier to the checksum. Increment the multiplier after each character.
4. Divide the result by 103. The remainder is the checksum.
5. Convert the numeric checksum into an ASCII character.

The details of converting the numeric checksum to an ASCII character may vary depending on the particular font being used. Special handling may be required in cases where the resulting ASCII character code equals zero or is above 95.

Generally speaking, in Subset A if the checksum is between 0 and 63 inclusive, add the checksum and the ASCII code for a space (32) to obtain the character code. If the checksum is 64 or higher, subtract 64 to obtain the character code. In Subset B, add the checksum and the ASCII code for a space (32). In Subset C, the checksum is the ASCII value of the character.

ABCxyz#$%15z

Document	**1 / 1**
Name	Document_48
Created	13. 10. 2017 11:53:16
Type	image/tiff
Size	699.53 KB
Page	**1 / 18**
Width	2480
Height	3507
Resolution	300 dpi
Color	Black & White (1-Bit White0)
BARCODE	
Text	123456789012
Type	Type-128
Position	(935, 829)
BARCODE	
Text	ABCxyz#$%15z
Type	Type-128
Position	(773, 2626)

Image 56: Example of using the classic sample of a user interface display

In the classic sample, the following components are used:

- imis.scan.ui.Thumbnails
- imis.scan.ui.ImageDetails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton

5.4.1.1 classic.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>IMiS/wScan Classic</title>
  <link rel="shortcut icon" type="image/png" href="img/favicon.png" />
  <link rel="stylesheet" href="imis.scan.ui.css" />
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="style/classic.css" />
</head>
<body>
  <div class="imis-scan-app">
    <nav id="nav-top">
      <div id="title" class="title"><a href="/">IMiS/wScan</a><div id="scan-
status"></div></div>
      <div id="imis-profile"></div>
    </nav>
    <nav id="nav" style="width: 500px; margin:auto;">
      <div id="scan-btn">Scan</div>
      <div id="continue-btn">Continue</div>
      <div id="cancel-btn">Cancel</div>
      <div id="download-btn">Save</div>
    </nav>
    <div id="imis-progress"></div>
    <div class="main" id="main">
      <div id="thumbnails" class="main-left">Thumbnails</div>
      <div id="images" class="main-center"></div>
      <div id="image-details" class="main-right">Details</div>
    </div>
  </div>
  <script src="imis.scan.js"></script>
  <script src="imis.scan.ui.js"></script>
<script>
  window.addEventListener('load', function () {

    // Set scan version to title attribute
    document.getElementById("title").setAttribute("title", imis.scan.ui.version);

    // Create image details object
    const imageDetailsUI = new imis.scan.ui.ImageDetails({
      id: "image-details",
      onClose: function () {
        // Hide details
        imageDetailsUI.hide();
        // Resize images
        document.getElementById("images").setAttribute("style", "flex: 1");
      }
    });

    try {
      // Create a scan object
      var scan = new imis.scan.ui.Scan({
        //url: "http://example.com",
        apiKey = API_KEY,
        thumbnails: new imis.scan.ui.Thumbnails({
          id: "thumbnails",
          orientation: "vertical",
          thumbnail: {
            width: 80, // Thumbnail width
            title: true
          },
          contextMenu: {
            enabled: true,
            properties: true,
            onPropertiesSelected: function () {
              // Show details
              imageDetailsUI.show();
              // Reset images style
              document.getElementById("images").setAttribute("style", "");
            }
          }
        }),
        imageDetails: imageDetailsUI,
        status: new imis.scan.ui.Status({ id: "scan-status" }),
        progress: new imis.scan.ui.Progress({ id: "imis-progress" }),
        images: new imis.scan.ui.ImageScroll({
          id: "images",
          controls: true,
          pageIndex: true,
          contextMenu: {
            onPropertiesSelected: function () {
              // Show details
              imageDetailsUI.show();
              // Reset images style
              document.getElementById("images").setAttribute("style", "");
            }
          }
        }),
        buttons: {
          scan: new imis.scan.ui.Button({ id: "scan-btn" }),
          profiles: new imis.scan.ui.ProfilesButton({id: "imis-profile" }),
          download: new imis.scan.ui.Button({ id: "download-btn" }),
          cancel: new imis.scan.ui.Button({ id: "cancel-btn" }),
          continue: new imis.scan.ui.Button({ id: "continue-btn" })
        },
        onError: function (message) {
          // Show dialog with error message
          new imis.scan.ui.AlertDialog({ title: "Error", text: message });
        }
      });
      scan.show();
      refreshLayoutHeight();
    } catch (e) {
      console.error(e);
    }
  });

  // Resizes height of main element
  function refreshLayoutHeight() {
    var titleHeight = document.getElementById("nav-top").offsetHeight;
    var navHeight = document.getElementById("nav").offsetHeight +
document.getElementById("imis-progress").offsetHeight;
    var mainHeight = (window.innerHeight - titleHeight - navHeight);
    document.getElementById("main").style.height = mainHeight + "px";
  }

  // Resize event listener
  window.addEventListener('resize', function (event) {
    refreshLayoutHeight();
  });
</script>
</body>
</html>
```

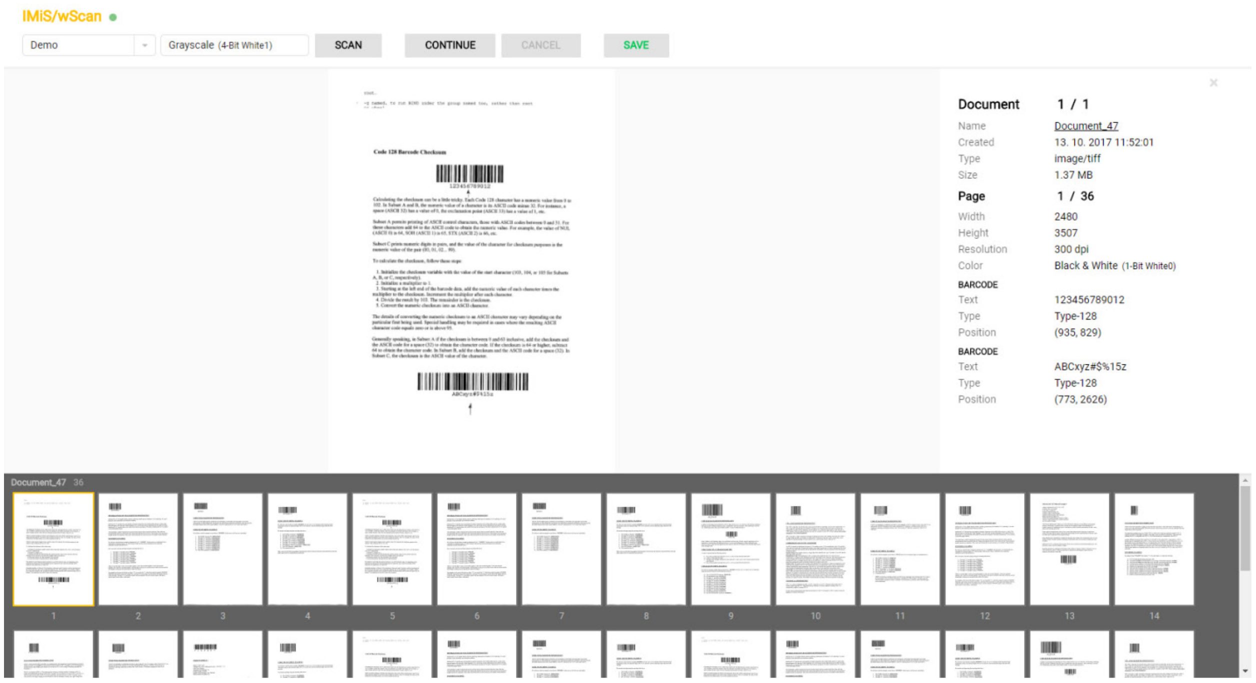
5.4.1.2 classic.css

```
body {
  margin: 0;
  background: #fff;
  height: 100%;
  width: 100%;
  font-family: 'Roboto', sans-serif;
}

a {
  text-decoration: none;
  color: inherit;
}

.title {
  background: #fff;
  color: #FFC107;
  font-size: 20px;
  padding-top: 15px;
  font-weight: bold;
}

nav {
  margin: 0;
  position: relative;
  padding-bottom: 10px;
  padding-left: 25px;
  padding-right: 25px;
  background: #fff;
}

nav > div {
  display: inline-block;
  margin-right: 2px;
}

.main {
  background: #EEEEEE;
  display: flex;
  height: 500px;
  position: relative;
  box-shadow: 0 0 2px 2px rgba(0,0,0,0.075);
  z-index: 1000;
}
```

```
.main-left,
.main-right {
  flex: 0.25;
}

.main-center {
  flex: 0.50;
}

#download-btn {
  margin-left: 25px;
}
```

5.4.2 Modern sample

Modern sample is suitable for application developers that keep up with the more recent trends in user interface appearance. Document pages take up most of the user interface. By default, page details are located on the right. A user can easily close or reopen them by selecting the menu on an individual page. For greater transparency the thumbnails of document pages are sorted at the bottom, which is especially suitable for larger volumes of scanned documents.



Document	**1 / 1**
Name	Document_47
Created	13. 10. 2017 11:52:01
Type	image/tiff
Size	1.37 MB
Page	**1 / 36**
Width	2480
Height	3507
Resolution	300 dpi
Color	Black & White (1-Bit White0)
BARCODE	
Text	123456789012
Type	Type-128
Position	(935, 829)
BARCODE	
Text	ABCxyz#$%15z
Type	Type-128
Position	(773, 2626)

Image 57: Example of using the modern sample of a user interface display

In the modern sample, the following components are used:

- imis.scan.ui.Thumbnails
- imis.scan.ui.ImageView
- imis.scan.ui.ImageDetails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton
- imis.scan.ui.ColorDropdownButton

5.4.2.1 modern.html

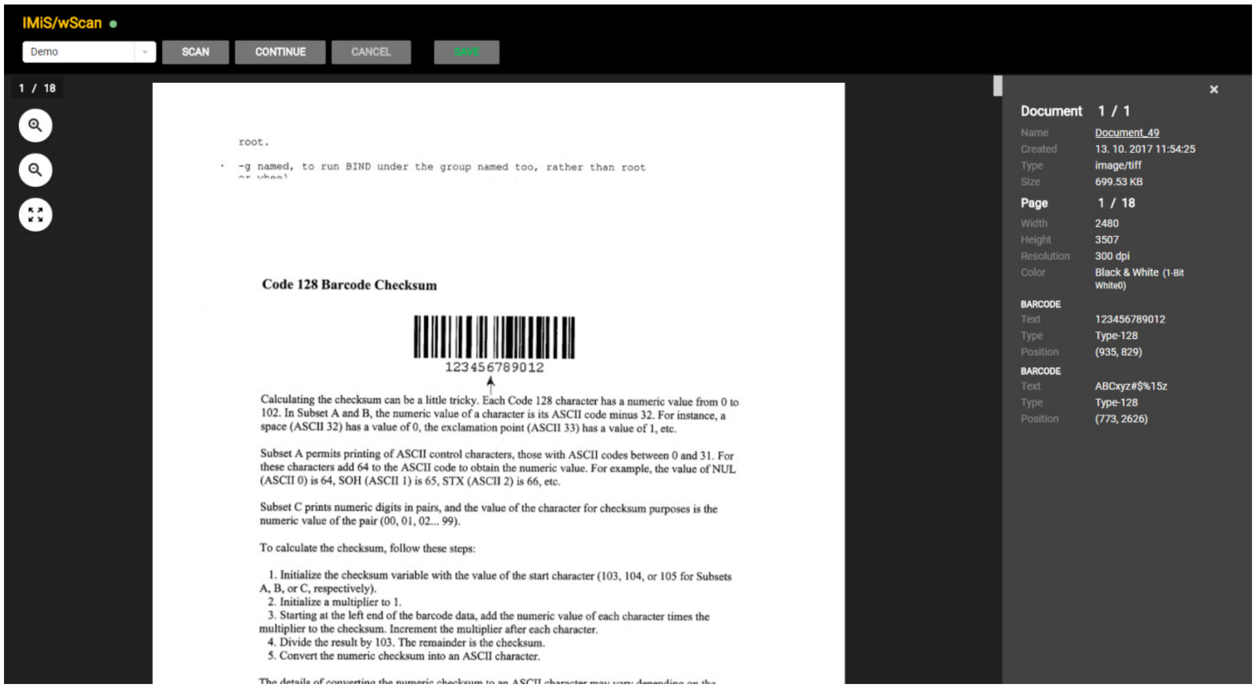
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>IMiS/wScan Modern</title>
  <link rel="shortcut icon" type="image/png" href="img/favicon.png" />
  <link rel="stylesheet" href="imis.scan.ui.css" />
  <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
  <link rel="stylesheet" href="style/modern.css" />
</head>
<body>
  <nav id="nav-top">
    <div id="title" class="title"><a href="/">IMiS/wScan</a><div id="scan-
status"></div></div>
  </nav>
  <nav id="nav">
    <div id="imis-profile"></div>
    <div id="imis-profile-color"></div>
    <div id="scan-btn"></div>
    <div id="continue-btn" style="margin-left:25px;"></div>
    <div id="cancel-btn"></div>
    <div id="download-btn"></div>
  </nav>
  <div id="imis-progress"></div>
  <div class="main" id="main">
    <div id="image-view" class="main-center"></div>
    <div id="image-details" class="main-right"></div>
  </div>
  <div id="thumbnails"></div>
  <script src="imis.scan.js"></script>
  <script src="imis.scan.ui.js"></script>
<script>
  window.addEventListener('load', function () {

    //imis.scan.ui.Resource.lang = "si";

    document.getElementById("title").setAttribute("title", imis.scan.ui.version);

    try {
      var scan = new imis.scan.ui.Scan({
        //url: "http://beno:5000",
        //url: "https://beno:5443",
        //url: "http://localhost:5000",
        //notifications: false,
        //useLocalStorage: false,
        apiKey = API_KEY,
        thumbnails: new imis.scan.ui.Thumbnails({
          id: "thumbnails",
          //backgroundColor: "#f5f5f5",
          //color: "#0f0",
          //focusNewPage: false,
          //orientation: "horizontal",
          thumbnail: {
            //width: 25,
            //height: 250,
            //title: false
            //titleColor: "#00f"
          },
          contextMenu: {
            enabled: true,
            onPropertiesSelected: function() {
              document.getElementById("image-details").style.display = null;
              document.getElementById("image-view").style.width = null;
            }
          }
        }),
        imageView: new imis.scan.ui.ImageView({
          id: "image-view",
          onPropertiesSelected: function () {
            document.getElementById("image-details").style.display = null;
            document.getElementById("image-view").style.width = null;
          }
        }),
        imageDetails: new imis.scan.ui.ImageDetails({
          id: "image-details",
          //background: "#000",
          //color: "#0f0",
          onClose: function () {
            document.getElementById("image-details").style.display = "none";
            document.getElementById("image-view").style.width = "100%";
          }
        }),
        status: new imis.scan.ui.Status({ id: "scan-status" }),
        progress: new imis.scan.ui.Progress({ id: "imis-progress" }),
        buttons: {
          scan: new imis.scan.ui.Button({ id: "scan-btn" }),
          profiles: new imis.scan.ui.ProfilesButton({ id: "imis-profile" }),
          download: new imis.scan.ui.Button({ id: "download-btn" }),
          cancel: new imis.scan.ui.Button({ id: "cancel-btn" }),
          continue: new imis.scan.ui.Button({ id: "continue-btn" }),
          color: new imis.scan.ui.ColorDropdownButton({ id: "imis-profile-color" })
        },
        onError: function (message) {
          new imis.scan.ui.AlertDialog({ title: "Error", text: message });
        }
      });
      scan.show();
      refreshLayoutHeight();
    } catch (e) {
      console.error(e);
      // Display error dialog
      new imis.scan.ui.AlertDialog({title: "Error", text: e});
    }
  });
  // Resize main and thumbnails height
  function refreshLayoutHeight() {
    var navHeight = document.getElementById("nav-top").offsetHeight +
      document.getElementById("nav").offsetHeight + document.getElementById("imis-
progress").offsetHeight;
    var thumbnailsHeight = window.innerHeight * 0.30; // 30% of window size
    // Main height
    document.getElementById("main").style.height = (window.innerHeight - navHeight -
thumbnailsHeight) + "px";
    // Thumbnails height
    document.getElementById("thumbnails").style.height = thumbnailsHeight + "px";
  }

  // Window resize event
  window.addEventListener('resize', function (event) {
    console.log("resize");
    refreshLayoutHeight();
  });
</script>
</body>
</html>
```

5.4.2.2 modern.css

```
body {
  margin: 0;
  background: #fff;
  height: 100%;
  width: 100%;
  font-family: 'Roboto', sans-serif;
}

a {
  text-decoration: none;
  color: inherit;
}

.title {
  background: #fff;
  color: #FFC107;
  font-size: 20px;
  padding-top: 15px;
  font-weight: bold;
}

nav {
  margin: 0;
  position: relative;
  padding-bottom: 10px;
  padding-left: 25px;
  padding-right: 25px;
  background: #fff;
}

nav > div {
  display: inline-block;
  margin-right: 2px;
}

.main {
  background: #EEEEEE;
  overflow: hidden;
  height: 500px;
  position: relative;
  box-shadow: 0 0 2px 2px rgba(0,0,0,0.075);
  z-index: 1000;
}

.main-left {
}

.main-center {
  float: left;
  width: 75%;
}

.main-right {
  float: left;
  width: 25%;
}

#download-btn {
  margin-left: 25px;
}

.imis-status {
  display: inline-block;
  margin-left: 10px;
}
```

5.4.3 Classic dark sample

Classic dark sample follows the latest trends of important document viewers. Document pages are shown in the center. The user navigates between pages with the slider. Documents are not visually separated from one another. It is clear from the page details to which documents the pages belong. By default, page details are not displayed. The user opens them by selecting the menu on an individual page. They are shown on the right side of the user interface.
Thumbnails of document pages are not available.



root.

-g named, to run BIND under the group named too, rather than root

Code 128 Barcode Checksum

123456789012

Calculating the checksum can be a little tricky. Each Code 128 character has a numeric value from 0 to 102. In Subset A and B, the numeric value of a character is its ASCII code minus 32. For instance, a space (ASCII 32) has a value of 0, the exclamation point (ASCII 33) has a value of 1, etc.

Subset A permits printing of ASCII control characters, those with ASCII codes between 0 and 31. For these characters add 64 to the ASCII code to obtain the numeric value. For example, the value of NUL (ASCII 0) is 64, SOH (ASCII 1) is 65, STX (ASCII 2) is 66, etc.

Subset C prints numeric digits in pairs, and the value of the character for checksum purposes is the numeric value of the pair (00, 01, 02... 99).

To calculate the checksum, follow these steps:

1. Initialize the checksum variable with the value of the start character (103, 104, or 105 for Subsets A, B, or C, respectively).
2. Initialize a multiplier to 1.
3. Starting at the left end of the barcode data, add the numeric value of each character times the multiplier to the checksum. Increment the multiplier after each character.
4. Divide the result by 103. The remainder is the checksum.
5. Convert the numeric checksum into an ASCII character.

Document	**1 / 1**
Name	Document_49
Created	13. 10. 2017 11:54:25
Type	image/tiff
Size	699.53 KB
Page	**1 / 18**
Width	2480
Height	3507
Resolution	300 dpi
Color	Black & White (1-Bit White0)
BARCODE	
Text	123456789012
Type	Type-128
Position	(935, 829)
BARCODE	
Text	ABCxyz#$%15z
Type	Type-128
Position	(773, 2626)

Image 58: Example of using the classic (dark) sample of a user interface display

In the classic (dark) sample, the following components are used:

- imis.scan.ui.ImageScroll
- imis.scan.ui.ImageDetails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton

5.4.3.1 classic_dark.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>IMiS/wScan Classic (dark)</title>
  <link rel="shortcut icon" type="image/png" href="img/favicon.png" />
  <link rel="stylesheet" href="imis.scan.ui.css" />
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="style/classic.dark.css" />
</head>
<body>
  <nav id="nav-top">
    <div id="title" class="title"><a href="/">IMiS/wScan</a><div id="scan-
status"></div></div>
  </nav>
  <nav id="nav">
    <div id="imis-profile"></div>
    <div id="scan-btn">Scan</div>
    <div id="continue-btn">Continue</div>
    <div id="cancel-btn">Cancel</div>
    <div id="download-btn">Save</div>
  </nav>
  <div id="imis-progress"></div>
  <div class="main" id="main">
    <div id="images" class="main-center"></div>
    <div id="image-details" class="main-right">Details</div>
  </div>
  <script src="imis.scan.js"></script>
  <script src="imis.scan.ui.js"></script>
<script>
  window.addEventListener('load', function () {

    // Set scan version to title attribute
    document.getElementById("title").setAttribute("title", imis.scan.ui.version);

    // Create image details object
    var imageDetailsUI = new imis.scan.ui.ImageDetails({
      id: "image-details",
      closed: true,
      darkMode: true,
      onClose: function () {
        // Hide details
        imageDetailsUI.hide();
        // Resize images
        document.getElementById("images").style.flex = 1;
      }
    });

    // Resize images
    document.getElementById("images").style.flex = 1;

    try {
      var scan = new imis.scan.ui.Scan({
        //url: "http://example.com",
        apiKey = API_KEY,
        imageDetails: imageDetailsUI,
        status: new imis.scan.ui.Status({ id: "scan-status" }),
        progress: new imis.scan.ui.Progress({ id: "imis-progress", darkMode: true }),
        images: new imis.scan.ui.ImageScroll({
          id: "images",
          //background: "#f5f5f5",
          darkMode: true,
          focusNewPage: false,
          contextMenu: {
            enabled: true,
            onPropertiesSelected: function () {
              imageDetailsUI.show();
              document.getElementById("images").style.flex = undefined;
            }
          }
        }),
        buttons: {
          scan: new imis.scan.ui.Button({ id: "scan-btn", darkMode: true }),
          profiles: new imis.scan.ui.ProfilesButton({ id: "imis-profile" }),
          download: new imis.scan.ui.Button({ id: "download-btn", darkMode: true }),
          cancel: new imis.scan.ui.Button({ id: "cancel-btn", darkMode: true }),
          continue: new imis.scan.ui.Button({ id: "continue-btn", darkMode: true })
        },
        onError: function (message) {
          // Show dialog with error message
          new imis.scan.ui.AlertDialog({ title: "Error", text: message });
        }
      });
      scan.show();
      refreshLayoutHeight();
    } catch (e) {
      console.error(e);
    }
  });

  // Resizes height of main element
  function refreshLayoutHeight() {
    var titleHeight = document.getElementById("nav-top").offsetHeight;
    var navHeight = document.getElementById("nav").offsetHeight +
document.getElementById("imis-progress").offsetHeight;
    var mainHeight = (window.innerHeight - titleHeight - navHeight);
    document.getElementById("main").style.height = mainHeight + "px";
  }

  // Resize event listener
  window.addEventListener('resize', function (event) {
    console.log("resize");
    refreshLayoutHeight();
  });
</script>
</body>
</html>
```

5.4.3.2 classic.dark.css

```
body {
  margin: 0;
  background: #fff;
  height: 100%;
  width: 100%;
  font-family: 'Roboto', sans-serif;
}

a {
  text-decoration: none;
  color: inherit;
}

.title {
  color: #FFC107;
  font-size: 20px;
  padding-top: 15px;
  font-weight: bold;
}

nav {
  margin: 0;
  position: relative;
  padding-bottom: 10px;
  padding-left: 25px;
  padding-right: 25px;
  background: #000;
}

nav > div {
  display: inline-block;
  margin-right: 2px;
}

.main {
  background: #EEEEEE;
  display: flex;
  height: 500px;
  position: relative;
  box-shadow: 0 0 2px 2px rgba(255,255,255,0.1);
  z-index: 1000;
}

.main-center {
  flex: 0.75;
}

.main-right {
  flex: 0.25;
}

#download-btn {
  margin-left: 25px;
}
```

5.4.4 Gallery sample

Gallery sample is suitable for displaying larger volumes of scanned documents or for batch scanning. Thumbnails of document pages are bigger and more visible than in the other samples. They are displayed in rows across the entire user interface. Documents are separated based on the number of pages specified in the settings. Page details appear to the user by double-clicking on an individual page.

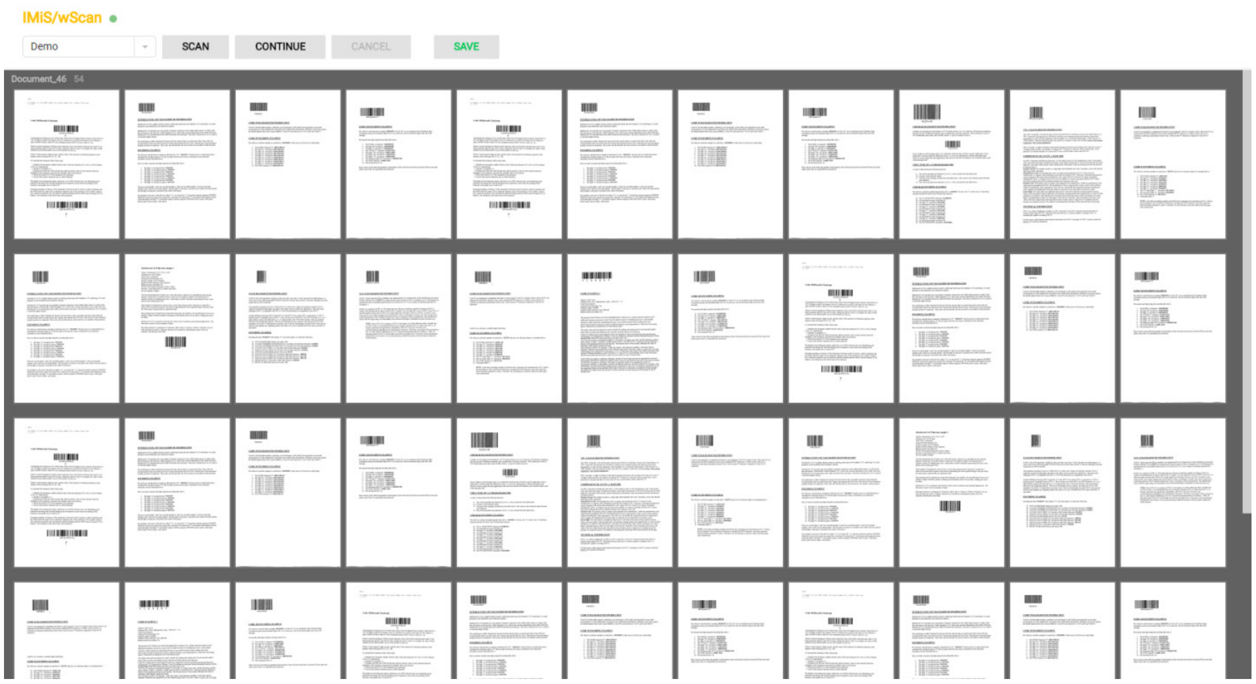


Image 59: Example of using the gallery sample of a user interface display

In the gallery sample, the following components are used:

- imis.scan.ui.Thumbnails
- imis.scan.ui.Status
- imis.scan.ui.Progress
- imis.scan.ui.Button
- imis.scan.ui.ProfilesButton

5.4.4.1 gallery.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>IMiS/wScan Gallery</title>
  <link rel="shortcut icon" type="image/png" href="img/favicon.png" />
  <link rel="stylesheet" href="imis.scan.ui.css" />
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto" />
  <link rel="stylesheet" href="style/gallery.css" />
</head>
<body>
  <nav id="nav-top">
    <div id="title" class="title"><a href="/">IMiS/wScan</a><div id="scan-
status"></div></div>
  </nav>
  <nav id="nav">
    <div id="imis-profile"></div>
    <div id="scan-btn">Scan</div>
    <div id="continue-btn">Continue</div>
    <div id="cancel-btn">Cancel</div>
    <div id="download-btn">Save</div>
  </nav>
  <div id="imis-progress"></div>
  <div class="main" id="main">
    <div id="thumbnails">Thumbnails</div>
  </div>
  <script src="imis.scan.js"></script>
  <script src="imis.scan.ui.js"></script>
<script>
  window.addEventListener('load', function () {

    // Set scan version to title attribute
    document.getElementById("title").setAttribute("title", imis.scan.ui.version);

    try {
      const scan = new imis.scan.ui.Scan({
        //url: "http://example.com",
        apiKey = API_KEY,
        thumbnails: new imis.scan.ui.Thumbnails({
          id: "thumbnails",
          //darkMode: false,
          orientation: "horizontal",
          thumbnail: {
            height: 200, // thumbnail height
            title: false
          },
          gallery: true,
          contextMenu: {
            enabled: false
          }
        }),
        status: new imis.scan.ui.Status({ id: "scan-status" }),
        progress: new imis.scan.ui.Progress({ id: "imis-progress" }),
        buttons: {
          scan: new imis.scan.ui.Button({ id: "scan-btn" }),
          profiles: new imis.scan.ui.ProfilesButton({ id: "imis-profile" }),
          download: new imis.scan.ui.Button({ id: "download-btn" }),
          cancel: new imis.scan.ui.Button({ id: "cancel-btn" }),
          continue: new imis.scan.ui.Button({ id: "continue-btn" })
        },
        onError: function (message) {
          // Show dialog with error message
          new imis.scan.ui.AlertDialog({ title: "Error", text: message });
        }
      });
      scan.show();
      refreshLayoutHeight();
    } catch (e) {
      console.error(e);
      // Show dialog with error message
      new imis.scan.ui.AlertDialog({ title: "Error", text: e });
    }
  });

  // Resizes height of main element
  function refreshLayoutHeight() {
    var titleHeight = document.getElementById("nav-top").offsetHeight;
    var navHeight = document.getElementById("nav").offsetHeight +
document.getElementById("imis-progress").offsetHeight;
```

```
    document.getElementById("main").style.height = (window.innerHeight - titleHeight -
navHeight) + "px";
  }

  // Resize event listener
  window.addEventListener('resize', function (event) {
    refreshLayoutHeight();
  });
</script>
</body>
</html>
```

5.4.4.2 gallery.css

```
body {
  margin: 0;
  background: #fff;
  height: 100%;
  width: 100%;
  font-family: 'Roboto', sans-serif;
}

a {
  text-decoration: none;
  color: inherit;
}

.title {
  background: #fff;
  color: #FFC107;
  font-size: 20px;
  padding-top: 15px;
  font-weight: bold;
}

nav {
  margin: 0;
  position: relative;
  padding-bottom: 10px;
  padding-left: 25px;
  padding-right: 25px;
  background: #fff;
}

nav > div {
  display: inline-block;
  margin-right: 2px;
}

.main {
  position: relative;
  z-index: 1000;
}

#download-btn {
  margin-left: 25px;
}
```

6 TROUBLESHOOTING

6.1 Problems using IMiS®/wScan

Below, issues frequently encountered when using IMiS®/wScan application are described and instructions for resolving them are given.

6.1.1 Error »Forbiden«

Cause of problem

The web application has not forwarded the correct security key via the »imis.scan.js« library to the IMiS®/Capture Service.

Solution for problem

The user with administrator authorization must verify that the security key is the same both in the IMiS®/Capture Service and the web application. If a new security key was entered or created in the IMiS®/Capture Service, a user with administrator authorization must restart the service. After the restart, the browser's web page for displaying scanned documents must be refreshed.

6.1.2 Error »Error in establishing connection«

Cause of problem

The web application accesses the IMiS®/Capture Service via a domain that is not allowed in the IMiS®/Capture Service.

Solution for problem

A user with administrator authorization must enter the allowed domains from which the web application can access the IMiS®/Capture Service. After entering them in the IMiS®/Capture Service, the service must be restarted. After the restart, the browser's web page for displaying scanned documents must be refreshed.

6.1.3 Error “Socket connection error”

Cause of problem

The web service IMiS®/Capture Service has not been started or the client does not have access rights for the address at which IMiS®/Capture Service is located.

Solution for problem

The administrator has to restart IMiS®/Capture Service. After restarting the service refresh the web page in the browser to show the scanned documents.

6.1.4 Error “No scanner is connected”

Cause of problem

When starting IMiS®/Capture Service, the service could not locate or load a driver for the connected scanner.

Solution for problem

The administrator has to check whether the scanner is turned on and connected to the computer. Next, check if the IMiS®/Scan application is running. If it is running, you have to stop it and restart IMiS®/Capture Service. For more information see chapter 4.2 Startup and closing.

6.1.5 Error “Scanner: ‘{scanner name}' cannot be loaded”.

Cause of problem

Prior to starting the scan IMiS®/Capture Service was unable to load a driver for the selected scanner.

Solution for problem

The administrator has to check whether the scanner is turned on and connected to the computer. Next, check
if the IMiS®/Scan application is running. If it is running, you have to stop it and try scanning in the IMiS®/wScan application. If scanning is still not working, you have to restart IMiS®/Capture Service. For more information see chapter 4.2 Startup and closing.

6.1.6 Scanning cannot be continued after successful scanning

Cause of problem

During scanning the data connection between the scanner and computer was terminated.

Solution for problem

The administrator has to turn the scanner off and disconnect it from the workstation.
Then turn the scanner back on and connect it to the workstation. Start the scanning procedure again in the IMiS®/wScan application. If scanning is still not working, open the Task Manager program. Find the process named "fjictwsw.exe" among all the processes and end it by clicking on the button "End process".

6.1.7 During scanning empty pages are not being removed

Cause of problem

In the profile settings the scanner settings for removing empty pages have not been configured.

Solution for problem

After configuring the profile in the IMiS®/wScan application, the administrator starts the IMiS®/wScan administration module and follows the procedures described in chapter 4.3. Additional settings.

6.1.8 Error "Your browser does not support Javascript ES6. Update browser."

Cause of problem

The "imis.scan.js" library does not work if the browser does not support the ECMAScript6 JavaScript standard.

Solution for problem

The existing browser has to be updated to the latest version.

Warning: The latest version of the Internet Explorer browser does not support this standard.

6.1.9 Error "Your browser does not support WebSockets. Update browser."

Cause of problem

The "imis.scan.js" library does not work if the browser does not support WebSocket technology.

Solution for problem

The existing browser has to be updated to the latest version.

Warning: The latest version of the Internet Explorer browser does not support this standard.