Plugins

April 26, 2022

Contents

Manatee from v1.28 has the ability to download and use new plugins and new modules.

A *plugin* is a component that can be started and stopped and may provide some functionality while it is running. An example is the Sectra plugin which interfaces with the Sectra application and provides a bridge between its built-in context manager and the context manager embedded in Manatee.

A *module* is a component that provides an API for use in flows. An example is the TOTP module that exposes functionality to generate time-based one-time passcodes for use in flows.

Modules and plugins are loaded as described in Modules where you can also read about how to query for available plugins/modules and their versions.

Audio module

The Audio module contains functionality to produce audio from the speakers of the machine on which it runs. It has the following main functionality:

- `Audio.say` get Manatee to speak a phrase
- `Audio.play` to play `.wav` file

Audio.say

The `say` function triggers Manatee to speak the phrase given.

```
var audio = Module.load("Audio", { version: "v1.2.0" });
audio.say("Hello, world!");
```

The default voice synthesizer is the built-in Windows Speech API (SAPI) which has english voices and renders the audio on the local machine. You can also choose to use the Google Text-to-Speech API which has support for multiple languages and much higher quality voice synthesis.

::: warning Google Text-to-Speech requires internet connectivity

Using Google TTS will send the phrase you want to speak to Google servers, so be aware of this if e.g. your phrase contains sensitive information.

:::

You can choose the Google TTS with the `synth` option - and you can then select a language with the `lang` option.

```
audio.say("Hej, verden!", { lang: 'da-DK', synth: 'Google' });
```

It is also possible to select a specific voice using the `voice` option. See Audio.voices for how to list available voices.

Audio.voices

Lists the available voices.

```
// List the default set of voices
var voices = audio.voices();
// List Danish Google voices
voices = audio.voices({ lang: "da-DK", synth: "Google" });
```

Audio.play

The `play` function will synchronously play the given `.wav` file.

```
audio.play("C:/Users/robot/Desktop/sound.wav");
```

Audio.volume

Controls the volume of the current audio device.

```
var v = audio.volume;
audio.volume = v + 1;
```

Audio.muted

Get/set the muted state of the current audio device.

```
if (audio.muted) audio.muted = false;
```

::: details Releases

v1.2.0 • Add `volume` and `muted` properties

v1.1.0 • Adds Google TTS.

v1.0.0 • Initial release with SAPI voices.

:::

TOTP module

The TOTP module can be used to generate one-time passcodes as specified in RFC6238. It uses the same algorithm as Google Authenticator etc so it should be compatible with the codes generated there. The use-cases it is meant for is e.g. using Manatee for logging into 2FA enabled sites and a concrete case was @lykke s Swedish RPA setup for logging into his machines using the Bomgar tool.

An example is given below;

```
// Load and instantiate the module
var totp = Module.load("Totp", { version: "v1.0.1" });
// You use the code given in the 2FA signup (where the QR code is also
    normally shown)
```

```
var passcode = totp.generate("G3PV4MGQ74S4NRWWBHCCFDS572ACXOC4");
```

where `G3PV4MGQ74S4NRWWBHCCFDS572ACXOC4` is the secret code given when you sign up for 2FA auth. This needs to be stored somewhere, e.g. directly in the flow or maybe encrypted in a Table and then used each time you need to generate a new passcode for login.

::: details Releases

v1.0.1* Initial release.

:::

Sectra plugin

The Sectra plugin (proper name is `SectraPlugin`) is a bridge between the context management features (currently only *CurrentUseCase*) and the context manager in Manatee. It is an `IRunnablePlugin` which means that it must be started and supplied with a configuration before it does anything.

The logic it runs when `start(...)` is invoked is something like:

```
while(not_stopped) {
  if(sectra.connect() && manetee.connect()) {
    keep_synchronized(sectra, manatee);
    wait(sectra.done || manatee.done);
  }
  wait(a bit);
}
```

Configuration

When `start(...)` is invoked a configuration object with the following properties can be given as argument.

```
{
  applicationIdentifier: '',                  // (required) The
     identifier of the ContextParticipant Sectra configuration to use (
     default is "eu.sirenia.plugins.sectra")
  applicationName: '',                        // (optional) The name of
      the application (default is "Sirenia Sectra Plugin")
```

```
4    defaultViewIdentifier: '',                  // (optional) Arg for
        Sectra context (default is "IDS7InfoOrMatrixWindow")
5    defaultAccessionNumberGroupId: '',          // (optional) Arg for
        Sectra context (default is "WISE Server std-pacs-12:7800")
6    defaultAccessionNumber: '',                 // (optional) Arg for
        Sectra context (default is null)
7    defaultExaminationNumber: '',               // (optional) Arg for
        Sectra context (default is null)
8    defaultPatientIdentifierIssuer: '',         // (optional) Arg for
        Sectra context (default is null)
9    pollingIntervalInSeconds: 5,                // (optional) How often
        do we check if we can connect to Sectra (default is 5)
10   completeSectraStateSubject: '',             // (optional) The subject
         of the context item which we should r/w the complete Sectra state
         as json (default is null, which means we won't synchronise the
        complete state)
11   patientIdentifierSubject: '',               // (optional) The subject
         for the patient identifier (default is null, no synch is done)
12   viewIdentifierSubject: '',                  // (optional) The subject
         for the view identifier (default is null, no synch is done)
13   examinationNumberSubject: '',               // (optional) The subject
         for the examination number (default is null, no synch is done)
14   accessionNumberSubject: '',                 // (optional) The subject
         for the accession number (default is null, no synch is done)
15   accessionNumberGroupIdSubject: '',          // (optional) The subject
         for the accession group id number (default is null, no synch is
        done)
16   forceRefresh: false,                        // (optional) Should we
        force-refresh Sectra on context changes (default is false)
17   maxSecondsToWaitForSectraTxCompletion: 10,  // (optional) How long do
         we wait for Sectra to complete an inbound tx (default is 10)
18   delayForInitialTxInSeconds: 2,              // (optional) How long to
         wait after Sectra connect to run initial sync (default is 10)
19   windowMatch: ''                             // (optional) Regex to
        match title of Sectra window to better determine when to start the
         inconsistency checker
20   maxWindowWaitInSeconds: 30                  // (optional) How many
        seconds to max wait for a matching window before starting the
        inconsistency checker
21 }
```

Examples

Start the plugin

```
Plugin.start(
  "SectraPlugin",
  "v0.5.10",
  {
    applicationIdentifier: "eu.sirenia.plugins.sectra",
    completeSectraStateSubject: "sectraState"
  }
);
```

Stop the plugin

```
Plugin.stop(
  "SectraPlugin",
  "v0.5.10"
);
```

Query the status of the plugin

```
Debug.ger(Plugin.status("SectraPlugin", "v0.5.10");
```

::: details Releases

v0.5.10 Initial (public) release.

:::

FlaUI Native Plugin

This plugin provides a newer alternative driver/application type for automation of native windows applications. The driver provided by this plugin is the same driver that is available as the default native driver built into Manatee v2.0+. For this reason, the FlauiNative plugin would generally only be of use in deployments of Manatee v1.28-1.29.

Example use cases

- When the automation you need to do is not possible in the original built-in native driver of Manatee up to v1.29.

- When you create new automation under Manatee v1.29 and you want to avoid using a plugin after upgrading Manatee to v2.0

This native driver is not fully compatible with fields and flows that were created with its predecessor. When switching existing fields and flows from the old to the new driver, thorough testing and some adaptation should be expected and planned for.

Loading the plugin

After Manatee has started up, it will need the plugin if applications are configured for it which require the new driver.

v1.29+

In v1.29+ there are two ways of automatically loading a plugin: 1. The Manatee setting `PreloadedPlugins` can be set to include the plugins to be loaded at startup. A possible value could be `FlaUINativePlugin v0.0.9`. There is sometimes some difficulty in managing manatee settings across a large organization if not all Manatees should run with the same settings. It may not be desirable for thousands of Manatees to load the plugin if only a handful of them need it. 2. A flow triggered to run after Manatee startup can load the plugin programmatically. For Manatee v1.29+ that might look as follows:

```
try {
  Plugin.start('FlaUINativePlugin', 'v0.0.9');
  Notification.show('started', 'Plugin started', 'FlaUINativePlugin v0
      .0.9', { timeout: 3 });
} catch(e) {
  Notification.show('fail', 'Plugin failed to start', e && (e.message
      || e));
}
```

v1.28

For Manatee v1.28 the only option is a flow loading the plugin - and a bit more code is required:

```
var version = '0.0.9';
try {
  Plugin.start('FlaUINativePlugin', 'v' + version, {
    '[eu.sirenia]CurrentWorkingDirectory': ('%appdata%\\Sirenia\\
       Manatee\\plugins\\FlaUINativePlugin\\' + version).toString(),
    MaxCacheHeatCount: Settings.Manatee.MaxCacheHeatCount,
```

```
    FieldCacheHeaterDelayInSeconds: Settings.Manatee.
        FieldCacheHeaterDelayInSeconds,
    MaxConcurrentTasksForWindowResolvers: Settings.Manatee.
        MaxConcurrentTasksForWindowResolvers || 20,
    WindowTextDeadlineInMs: Settings.Manatee.WindowTextDeadlineInMs,
    DefaultFieldCacheExpiryInSeconds: Settings.Manatee.
        DefaultFieldCacheExpiryInSeconds,
    DisableProcessCache: Settings.Manatee.DisableProcessCache,
    NativeAutomationTransactionTimeoutTimeoutInSeconds: Settings.
        Manatee.NativeAutomationTransactionTimeoutTimeoutInSeconds || 60
  });
  Notification.show('started', 'Plugin started', 'FlaUINativePlugin v'
      + version, { timeout: 3 });
} catch(e) {
  Notification.show('fail', 'Plugin failed to start', e && (e.message
      || e));
}
```

It is a good idea to keep the plugin loading flow in a headless app, which is then associated with the machines in need of the plugin along with the flow itself. ### Selecting the driver in Cuesta The driver provided by the FlaUI plugin can be selected like any other driver in the Cuesta app configuration page. It is shown with the name `FlaUI Native`.

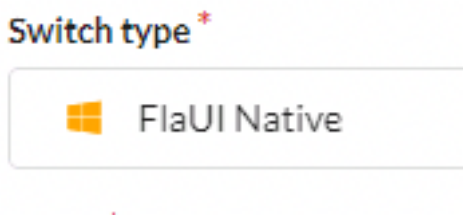

Figure 1: FlaUI selected in Cuesta

Keep in mind that such an app configuration is only going to work on Manatees that have the plugin running.

Registering the driver

If Cuesta doesn't offer the FlaUI plugin driver as an option it may be because the driver hasn't yet been registered. This is done by starting the plugin in Manatee (eg from a flow), then finding the plugin in the right click Manatee menu:

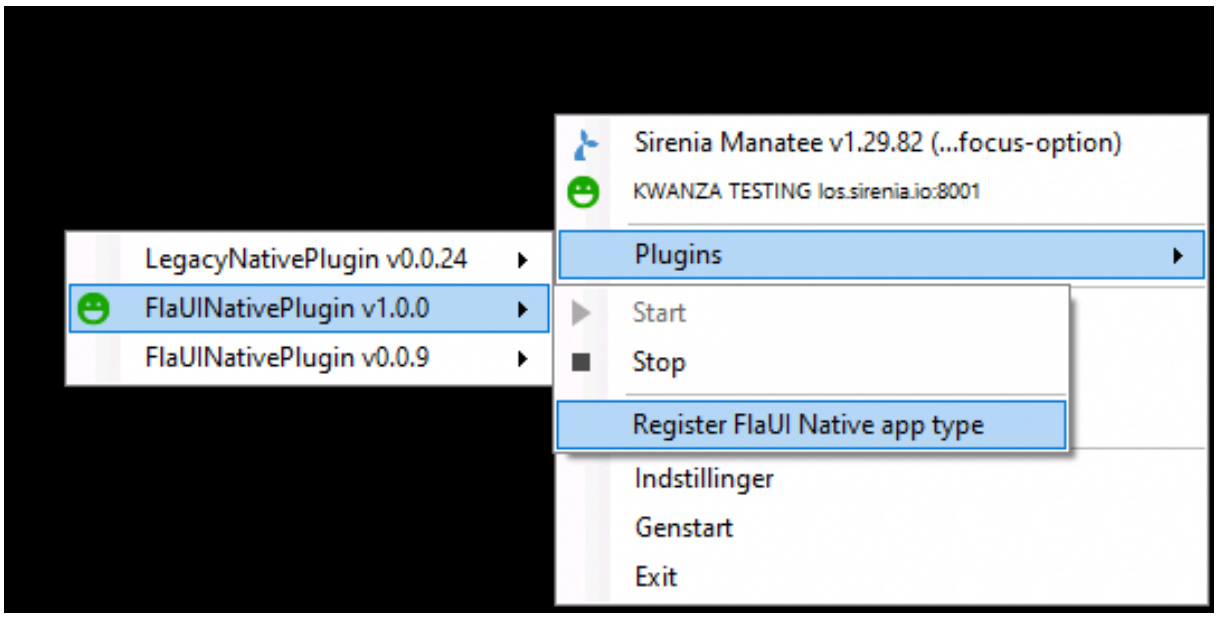

Figure 2: Registering a plugin driver

This only needs to be done once for Cuesta to know about the new available driver. At this point, it will be possible on the application settings page in Cuesta to choose the app type provided by the plugin.

Known differences with older driver

The two native driver implementations mostly work the same but there are some known differences that may cause issues if automation created on one driver is switched over to the other. A few concrete differences are described below, but there may be more since the implementation of eg the various clicks, input, select and so on are achieved by different means than before.

Path interpretation

Path traversal when resolving fields has been changed in subtle ways. The aim is to align the path semantics in the native driver with that of the other drivers (eg java and browser drivers). This concerns the interpretation of the `/` level divider in a path. With the old native driver, the path `Foo/Bar` would search the entire sub tree under `Foo` for at match to `Bar`. The new driver will only search among the immediate children of `Foo` for a match to `Bar`. To look for `Bar` in the full sub tree under `Foo`, use `Foo/**/Bar`.

Inspect content

The specific properties in the objects returned by calls to the `.inspect()` method change somewhat between the drivers due to differences in the data provided by the underlying windows automation layers.

::: details Releases

v0.0.9 Initial (public) release.

:::

Legacy Native Plugin

This plugin provides a driver compatible with the original native driver built into Manatee up to and including v1.29.

It mainly serves to provide backwards compatibility for existing native applications when Manatee is upgraded from v1.29 to v2.0. At the 2.0 level, native applications will start using this plugin automatically since the Manatee built-in driver was changed for the 2.0 release. ### Example use cases - When you run Manatee 2.0 and your flows and fields were made for the older native driver - In the unexpected event that some specific piece of automation works better under the old driver

Manatee 2.0

Upon the rollout of Manatee 2.0, the default built-in native driver changes to the new FlaUI driver. In order to avoid breaking existing native app automations when this happens, the responsibility of handling such apps is handed over to the LegacyNative plugin. Make sure to take steps to ensure that the LegacyNative plugin is running on Manatees that need it for running the existing native app automation.

In Cuesta 1.14 and Cuesta 1.15, this same driver is simply called `Native` as it is the default driver for native Windows applications up to and including Manatee v1.29.

In Cuesta 1.16, it is shown as `Legacy Native` to indicate that as of Manatee v2.0, it is no longer the recommended option for new Windows applications to be automated.

::: details Releases

v0.0.25 Initial (public) release.

:::

Pdf module

This module can be used to extract text from PDF files.

Note: That it is not possible to extract text from scanned PDF files.

Extract text from PDF file

You can use the `textBlocks` function to extract text blocks from a PDF file.

```
// Load and instantiate the module
var pdf = Module.load("Pdf", { version: "v1.0.4" }); // or choose a more recent version if one exists
var result = pdf.textBlocks("/path/to/file.pdf", { page: 2 });
// result is an `TextBlocks` object which has a blocks property containing the extracted text blocks
var blocks = result.blocks;
```

The options argument (the 2nd argument) can contain the following properties:

- `page` the page number to extract text from. Defaults to all pages.
- `password` the password to decrypt the PDF file.

The `TextBlocks` object has a `find` method which can be used to find text blocks in the PDF file. You can use it like:

```
// Find a block of text below the headline "Some headline"
result.find("below", "Some headline");
```

You can use

- `pdf.Below` aka `"Below"`
- `pdf.Above` aka `"Above"`
- `pdf.LeftOf` aka `"LeftOf"`
- `pdf.RightOf` aka `"RightOf"`
- `pdf.Nearest` aka `"Nearest"` to find the nearest block of text

as the first argument and a regular expression as the second argument.

`TextBlocks` also has a `blocks` property which contains the extracted text blocks. Each `TextBlock` object has the following properties:

- `text` (string) the combined text of all lines in the block
- `lines` (array of strings) the individual lines in the block
- `separator` (string) the separator used to separate the lines in the block
- `boundingBox` (object) the bounding box of the block
  - `topLeft` (number) the coordinate of the top left corner of the bounding box
  - `topRight` (number) the coordinate of the top right corner of the bounding box

  - `bottomLeft` (number) the coordinate of the bottom left corner of the bounding box
  - `bottomRight` (number) the coordinate of the bottom right corner of the bounding box
  - `width` (number) the width of the bounding box
  - `height` (number) the height of the bounding box
- `readingOrder` (number) the reading order of the block
- `textOrientation` (number) the text orientation of the block

::: details Releases

v1.0.3 Feature: Added support for password protected PDF files
v1.0.2 Initial release.

:::

Kant Plugin

This plugin provides an embedded Edge browser along with an associated app type so web pages can be automated. It mostly works the same as the Legacy Hosted Chrome app as well as the Krom plugin which also wraps a chrome browser.

This plugin is supported in Manatee v1.28+.

System requirements

This plugin requires the WebView2 Runtime to be installed. The installer comes as a free download from Microsoft. See here

Setting up for automation in Manatee 1.28.113

There are a number of steps involved in order to successfully launch and attach a web app using this plugin. They are summarized in the following checklist:

Use the plugin version 1.0.1+ for this

1. Identify set of groups for machines that should have the plugin (GROUPS)
2. Identify headless app (HEADLESS_APP) that can be used for starting the plugin and the browser
3. Create headless flow (PLUGIN_START) for starting the plugin
4. Set GROUPS on HEADLESS_APP and PLUGIN_START
5. Add a trigger to run PLUGIN_START on Manatee startup
6. Actually run PLUGIN_START
7. Register hosted edge plugin app type in cuesta via a manatee with the running plugin (if the app type isn't already made available)

8. Create headless flow (EDGE_START) in HEADLESS_APP for launching the browser (and reload Cuesta)
9. Add `url` input parameter to EDGE_START flow
10. Set up a Manatee Boot trigger on PLUGIN_START
11. Create plugin-edge-app (PLUGIN_EDGE_APP)
12. Set GROUPS on PLUGIN_EDGE_APP
13. Set `Launch With` to something like `manatee:runflow/withname/PLUGIN_START?url=https:%3A%2F%2Fdr.dk`
14. Set `Executable-path contains` to `Kant.exe`
15. Set `Title bar contains` to the launch url

PLUGIN_START flow

This flow gets run after Manatee startup and is responsible for starting the plugin. Note that it references the plugin version, so when updating the plugin, this flow needs to change.

```
var kantVersion = '1.0.1';
try {
  Plugin.start('KantPlugin', 'v' + kantVersion, {
    '[eu.sirenia]CurrentWorkingDirectory': ('%appdata%\\Sirenia\\
   Manatee\\plugins\\KantPlugin\\' + kantVersion).toString()
  });
  //Notification.show('kant', 'Plugin started', 'KantPlugin v' +
    kantVersion, { timeout: 3 });
} catch(e) {
  Notification.show('kant', 'Plugin failed to start', e && (e.message
    || e) + '\nv' + kantVersion, { severity: 'warn', timeout: 20 });
}
```

EDGE_START flow

This flow is used when Manatee needs to start a new instance of the hosted edge browser, Kant. Note that this flow also references the version of the plugin.

```
var kantBrowser = Module.load('KantBrowser', { version: '1.0.1' });

kantBrowser.launch(Inputs.url, {});
```

Setting up for automation in Manatee 1.28.121+

Use the plugin version 2.0.0+ for this

1. Identify set of groups for machines that should have the plugin (GROUPS)
2. Identify headless app (HEADLESS_APP) that can be used for starting the plugin
3. Create headless flow (PLUGIN_START) for starting the plugin
4. Set GROUPS on HEADLESS_APP and PLUGIN_START

5. Add a trigger to run PLUGIN_START on Manatee startup
6. Actually run PLUGIN_START
7. Register hosted edge plugin app type in cuesta via a manatee with the running plugin (skip this if the app type is already available)
8. Set up a Manatee Boot trigger on PLUGIN_START
9. Create plugin-edge-app (PLUGIN_EDGE_APP)
10. Set GROUPS on PLUGIN_EDGE_APP
11. Set `Launch With` to something like `{{appdata}}\\Sirenia\\Manatee\\plugins\\KantPlugin\\2.0.0\\Kant\\Kant.exe https://dr.dk` (replace the version and url to suit your purposes)
12. Set `Executable-path contains` to `Kant.exe`
13. Set `Title bar contains` to the url you wish to match on