Простой пример вызова функций плагина

Google Chrome/Chromium

Если вы открываете локальную страницу, убедитесь, что в настройках расширения Адаптер Рутокен Плагин включена опция **"Разрешить открывать локальные файлы по ссылкам"**

```
 - - - - - - - - - - - - - - -  - - - - - - - - - - - - - - - - - - - - - - -
|                                                                            |
         .    ra.rutoken.ru
|                                                                            |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


 ...
```

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="UTF-8">

<script type="text/javascript" src="rutoken.js"></script>
 <script type="text/javascript" src="scripts.js"></script>

<title>Rutoken Plugin Tutorial</title>
 </head>
 <body>
 <pre>
 - - - - - - - - - - - - - - -  - - - - - - - - - - - - - - - - - - - - - - -
| |
         . <a href="http://ra.rutoken.ru">   ra.rutoken.ru</a>
| |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 </pre>
 <div id="pluginStatus"><pre> ...</pre></div>
 <textarea rows="10" cols="95" id="textToSign" placeholder="   "></textarea><br/>
 <button id="signButton" onclick="sign()" style="display: block; background: none; border: 4px solid black; font-
family: monospace; width: 20em; height: 3em; text-align: center; cursor: pointer;"> </button>
 </body>
</html>

//   GOOGLE CHROME
//
//         ,      ,     .
//  -     .
//     ,       .
//    :
// promise.then(onFulfilled, onRejected)
// onFulfilled – ,          .
// onRejected – ,          .
//
//
//            "     "
var plugin;

function checkVersion(lastVersion) {
if (plugin.version.toString() < lastVersion)
console.log("download last version: " + lastVersion);
else
console.log("you have last version");
}

function getLastRtPluginVersion(callback) {
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://download.rutoken.ru/Rutoken_Plugin/Current/version.txt', true);
xhr.onreadystatechange = function() {
if (xhr.readyState == 4 && xhr.status == 200) {
var lastPluginVersion = this.response.split('Version: v.')[1].split('Release')[0].replace(/\s+/g, '');
```

```
callback(lastPluginVersion);
}
};
xhr.send();
}

window.onload = function() {
rutoken.ready
//    '  '  Google Chrome
.then(function() {
if (window.chrome || typeof InstallTrigger !== 'undefined') {
return rutoken.isExtensionInstalled();
} else {
return Promise.resolve(true);
}
})
//
.then(function(result) {
if (result) {
return rutoken.isPluginInstalled();
} else {
return Promise.reject("    '  '");
}
})
//
.then(function(result) {
if (result) {
return rutoken.loadPlugin();
} else {
return Promise.reject("   ");
}
})
//
.then(function(result) {
if (!result) {
return Promise.reject("   ");
} else {
plugin = result;
return Promise.resolve();
}
})
.then(function() {
document.getElementById("pluginStatus").innerHTML = "<pre> </pre>";
getLastRtPluginVersion(checkVersion);
}, function(msg) {
document.getElementById("pluginStatus").innerHTML = "<pre>" + msg + "</pre>";
});
}

//
//       https://dev.rutoken.ru/display/PUB/RutokenPluginDoc
//   CLASS: ERRORCODES
function handleError(reason) {
if (isNaN(reason.message)) {
alert(reason);
} else {
var errorCodes = plugin.errorCodes;
switch (parseInt(reason.message)) {
case errorCodes.PIN_INCORRECT:
alert(" PIN");
break;
default:
alert(" ");
}
}
}


sign = function() {
var rutokenHandle, certHandle;
//
var textToSign = document.getElementById("textToSign").value;
if (textToSign.length == 0) {
alert("    ");
return;
}
//
plugin.enumerateDevices()
.then(function(devices) {
if (devices.length > 0) {
return Promise.resolve(devices[0]);
} else {
return Promise.reject("  ");
}
```

```
})
//
.then(function(firstDevice) {
rutokenHandle = firstDevice;
return plugin.getDeviceInfo(rutokenHandle, plugin.TOKEN_INFO_IS_LOGGED_IN);
})
//        PIN-
.then(function(isLoggedIn) {
if (isLoggedIn) {
return Promise.resolve();
} else {
return plugin.login(rutokenHandle, "12345678");
}
})
//
.then(function() {
return plugin.enumerateCertificates(rutokenHandle, plugin.CERT_CATEGORY_USER);
})
//
.then(function(certs) {
if (certs.length > 0) {
certHandle = certs[0];
var options = {};
return plugin.sign(rutokenHandle, certHandle, textToSign, plugin.DATA_FORMAT_PLAIN, options);
} else {
return Promise.reject("    ");
}
})
//      CMS
.then(function(cms) {
alert(cms);
})
//
.then(function() {
plugin.logout(rutokenHandle);
}, handleError);
}


var rutoken = (function (my) {
 var loadCallbacks = [];
 var pluginMimeType = "application/x-rutoken-pki";
 var extension = window["C3B7563B-BF85-45B7-88FC-7CFF1BD3C2DB"];

function isFunction (obj) {
 return !!(obj && obj.call && obj.apply);
 }

function proxyMember (target, member) {
 if (isFunction(target[member])) {
 return function () {
 return target[member].apply(target, arguments);
 };
 } else {
 return target[member];
 }
 }

function returnPromise (promise) {
 return function () {
 return promise;
 };
 }

function initialize () {
 my.ready = Promise.resolve(true);
 my.isExtensionInstalled = returnPromise(Promise.resolve(false));
 my.isPluginInstalled = returnPromise(Promise.resolve(true));
 my.loadPlugin = loadPlugin;
 window.rutokenLoaded = onPluginLoaded;
 }

function initializeExtension() {
 var readyPromise = extension.initialize().then(function () {
 return extension.isPluginInstalled();
 }).then(function (result) {
 my.isExtensionInstalled = returnPromise(Promise.resolve(true));
 my.isPluginInstalled = proxyMember(extension, "isPluginInstalled");

if (result) {
 pluginMimeType = "application/x-rutoken-plugin";
 my.loadPlugin = loadChromePlugin;
 }
```

```
return true;
 });

my.ready = readyPromise;
 }

function initializeWithoutPlugin () {
 my.ready = Promise.resolve(true);
 my.isExtensionInstalled = returnPromise(Promise.resolve(false));
 my.isPluginInstalled = returnPromise(Promise.resolve(false));
 }

function loadPlugin () {
 var obj = document.createElement("object");
 obj.style.setProperty("visibility", "hidden", "important");
 obj.style.setProperty("width", "0px", "important");
 obj.style.setProperty("height", "0px", "important");
 obj.style.setProperty("margin", "0px", "important");
 obj.style.setProperty("padding", "0px", "important");
 obj.style.setProperty("border-style", "none", "important");
 obj.style.setProperty("border-width", "0px", "important");
 obj.style.setProperty("max-width", "0px", "important");
 obj.style.setProperty("max-height", "0px", "important");

// onload callback must be set before type attribute in IE earlier than 11.
 obj.innerHTML = "<param name='onload' value='rutokenLoaded'/>";
 // Just after setting type attribute before function returns promise
 // FireBreath uses onload callback to execute it with a small delay.
 // So it must be valid, but it will be called a little bit later.
 // In other browsers plugin will be initialized only after appending
 // an element to the document.
 obj.setAttribute("type", pluginMimeType);

document.body.appendChild(obj);

var promise = new Promise(function (resolve, reject) {
 loadCallbacks.push(resolve);
 });

return promise;
 }

function loadChromePlugin () {
 return extension.loadPlugin().then(function (plugin) {
 return resolveObject(plugin);
 }).then(function (resolvedPlugin) {
 resolvedPlugin.wrapWithOldInterface = wrapNewPluginWithOldInterface;
 return resolvedPlugin;
 });
 }

function onPluginLoaded (plugin, error) {
 wrapOldPluginWithNewInterface(plugin).then(function (wrappedPlugin) {
 if (loadCallbacks.length == 0) {
 throw "Internal error";
 }

var callback = loadCallbacks.shift();
 callback(wrappedPlugin);
 });
 }

function resolveObject (obj) {
 var resolvedObject = {};
 var promises = [];

for (var m in obj) {
 (function (m) {
 if (isFunction(obj[m].then)) {
 promises.push(obj[m].then(function (result) {
 return resolveObject(result).then(function (resolvedProperty) {
 if (isFunction(resolvedProperty)) {
 resolvedObject[m] = proxyMember(obj, m);
 } else {
 resolvedObject[m] = resolvedProperty;
 }
 });
 }));
 } else {
 resolvedObject[m] = obj[m];
 }
 })(m);
 }
```

```
if (promises.length == 0) {
 return new Promise(function (resolve) {
 resolve(obj);
 });
 } else {
 return Promise.all(promises).then(function () {
 return resolvedObject;
 });
 }
 }

function wrapNewPluginWithOldInterface () {
 var wrappedPlugin = {};

for (var m in this) {
 if (isFunction(this[m])) {
 wrappedPlugin[m] = (function(plugin, member) {
 return function() {
 var successCallback = arguments[arguments.length - 2];
 var errorCallback = arguments[arguments.length - 1];
 var args = Array.prototype.slice.call(arguments, 0, -2);
 return member.apply(plugin, args).then(function (result) {
 successCallback(result);
 }, function (error) {
 errorCallback(error.message);
 });
 };
 })(this, this[m]);
 } else {
 wrappedPlugin[m] = this[m];
 }
 }

return new Promise(function (resolve) {
 resolve(wrappedPlugin);
 });
 }

function wrapOldPluginWithOldInterface () {
 var unwrappedPlugin = { originalObject: this.originalObject };

 for (var m in this.originalObject) {
 unwrappedPlugin[m] = proxyMember(this.originalObject, m);
 }

return new Promise(function (resolve) {
 resolve(unwrappedPlugin);
 });
 }

function wrapOldPluginWithNewInterface (plugin) {
 var wrappedPlugin = {
 originalObject: plugin,
 wrapWithOldInterface: wrapOldPluginWithOldInterface
 };

for (var m in plugin) {
 if (isFunction(plugin[m])) {
 wrappedPlugin[m] = (function (plugin, member) {
 return function() {
 var args = Array.prototype.slice.call(arguments);
 return new Promise(function (resolve, reject) {
 args.push(resolve, reject);
 member.apply(plugin, args);
 });
 };
 })(plugin, plugin[m]);
 } else {
 wrappedPlugin[m] = plugin[m];
 }
 }

return new Promise(function (resolve) {
 resolve(wrappedPlugin);
 });
 }

if (extension) {
 initializeExtension();
 } else if (navigator.mimeTypes && navigator.mimeTypes[pluginMimeType]) {
 initialize();
 } else {
 try {
 var plugin = new ActiveXObject("Aktiv.CryptoPlugin");
```

```
 initialize();
 } catch (e) {
 initializeWithoutPlugin();
 }
 }

return my;
}(rutoken || {}));

if (typeof module !== 'undefined') {
 module.exports = rutoken;
}
```