API-документация для разработчиков

- **pkcs10** - Плагин для генерации ключевой пары и формирования запроса на сертификат формата PKCS#10
  - **create_pkcs10_from_key** - Формировать запрос на сертификат формата PKCS#10 из существующего ключа
  - **generate_keypair** - Сгенерировать ключевую пару
  - **create_pkcs10** - Формировать запрос на сертификат формата PKCS#10
  - **get_pkcs10_info** - Получить информацию о запросе PKCS#10

- **x509** - Плагин для работы с сертификатами X.509
  - **verify_certificate** - Верификация подписи сертификата субъектка сертификатом издателя
  - **get_certificate_info** - Получить информацию о сертификате
  - **get_certificate_chain** - Получить цепочку сертификатов в кодировке BASE64 по идентификатору ключа

- **truststore** - Плагин для работы с хранилищами доверенных сертификатов (ЗАГЛУШКА)
  - **list_truststore** - Получить список доверенных сертификатов (ЗАГЛУШКА)

- **ftjc** - Плагин для работы с USB-токеном FT Javacard (ЗАГЛУШКА)
  - **set_name** - Установить название USB-токену
  - **unload_key** - Удалить загруженные ключи по идентификатору
  - **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время
  - **store_certificates** - Сохранить сертификаты в USB-токен
  - **list_tokens** - Получить список токенов
  - **list_all_keys** - Получить список всех ключей из подключенных токенов
  - **install_applet** - Загрузить апплет в USB-токен
  - **get_user_data** - Извлечь пользовательские данные
  - **change_pin** - Изменить ПИН код токена
  - **verify_pin** - Проверить ПИН код токена
  - **get_random_data** - Создать случайные данные в USB-токене
  - **set_user_data** - Записать или Удалить пользовательские данные

- **ytks** - Плагин для работы с файлами хранилища ключей формата YTKS
  - **verify_password** - Проверить пароль хранилища ключей
  - **list_certificates** - Получить список сертификатов пользователя
  - **list_disks** - Получить список дисков
  - **change_password** - Изменить пароль хранилища ключей
  - **unload_key** - Удалить загруженные ключи по идентификатору
  - **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время
  - **list_all_certificates** - Получить список всех сертификатов пользователя
  - **save_ytks** - Сохранить ключевую пару или существующий ключ и новые сертификаты в новый файл формата YTKS

- **cipher** - Плагин для шифрования и дешифрования документа по алг.шифрования ГОСТ-28147, алг.обмена ключа ECDH-SHA256 в режиме P2P
  - **decrypt_document** - Дешифровать зашифрованный документ
  - **encrypt_document** - Создать зашифрованный документ

- **idcard** - Плагин для работы с ID-card E-IMZO
  - **verify_password** - Проверить пароль хранилища ключей (заглушка)
  - **personalize** - Персонализировать ID-карту записав новые сертификаты и установив PIN-код
  - **list_readers** - Получить список считывателей
  - **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время (заглушка)
  - **get_encrypted_signed_cplc** - Получить зашифрованный и подписанный заводской номер USB-токена
  - **list_all_certificates** - Получить список всех сертификатов пользователя (заглушка)

- **fileio** - Плагин для работы с файлами
  - **load_file** - Загруить файл

    - **write_file** - Записать содержимое zip файла на диск

- **tunnel** - Плагин для установки зашифрованное соединения с сервером по алг.шифрования ГОСТ-28147
    - **create_tunnel** - Создать зашифрованного соединения с сервером и вернуть TCP-порт для приема/передачи данных

- **pki** - Плагин для взаимодействия с ИОК
    - **enroll_pfx_step2** - Шаг №2 для получения ключа PFX
    - **enroll_pfx_step1** - Шаг №1 для получения ключа PFX

- **pkcs7** - Плагин для работы с форматом PKCS#7/CMS
    - **create_pkcs7** - Создать PKCS#7/CMS документ подписав ключем задаваемым идентификатором
    - **attach_timestamp_token_pkcs7** - Прикрепить токен штампа времени к документу PKCS#7/CMS (ЗАГЛУШКА)
    - **verify_pkcs7_attached** - Верифицировать документ PKCS#7/CMS, который содержит исходный документ (ЗАГЛУШКА)
    - **verify_pkcs7_detached** - Верифицировать документ PKCS#7/CMS, который не содержит исходный документ (ЗАГЛУШКА)
    - **verify_pkcs7_detached_crl** - Верифицировать документ PKCS#7/CMS, который не содержит исходный документ (ЗАГЛУШКА)
    - **append_pkcs7_attached** - Добавить подпись в существующий документ PKCS#7/CMS (DEPRECATED)
    - **get_pkcs7_detached_info** - Получить полную информацию о документе PKCS#7/CMS, который не содержит исходный документ
    - **verify_pkcs7_attached_crl** - Верифицировать документ PKCS#7/CMS, который содержит исходный документ (ЗАГЛУШКА)
    - **append_pkcs7_detached** - Добавить подпись в существующий документ PKCS#7/CMS, который не содержит исходного документа (DEPRECATED)
    - **get_pkcs7_attached_info** - Получить полную информацию о документе PKCS#7/CMS, который содержит исходный документ

- **cryptoauth** - Плагин для выполнения низкоуровневых криптографических преобразований (ЗАГЛУШКА)
    - **get_signature** - Подписать данные ключем задаваемым идентификатором
    - **get_digest_hex** - Вычислить хеш (в формате HEX) данных по алгоритму OZDST-1106-2009-2-A
    - **verify_digest_hex_signature_with_id** - Верифицировать подпись хеша (в формате HEX) ключем задаваемым идентификатором
    - **verify_signature_with_certificate** - Верифицировать подпись данных сертификатом
    - **get_digest_hex_signature** - Подписать хеш (в формате HEX) ключем задаваемым идентификатором
    - **verify_signature_with_id** - Верифицировать подпись данных ключем задаваемым идентификатором
    - **verify_digest_hex_signature_with_certificate** - Верифицировать подпись хеша (в формате HEX) сертификатом

- **certkey** - Плагин для работы с электронными ключами и сертификатами (ЗАГЛУШКА)
    - **unload_key** - Удалить загруженные ключи по идентификатору
    - **list_certificates** - Получить список сертификатов пользователя
    - **list_disks** - Получить список дисков
    - **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время
    - **list_all_certificates** - Получить список всех сертификатов пользователя

- **tsaclient** - Плагин для работы с токенами штампов времени
    - **get_timestamp_token_for_signature** - Получить токен штампа времени на подпись от службы штампов времени по веб-ссылке
    - **get_timestamp_token_request_for_data** - Получить запрос на получения токена штампа времени для данных
    - **get_timestamp_token_for_data** - Получить токен штампа времени на данные от службы штампов времени по веб-ссылке
    - **get_timestamp_token_info** - Получить информацию о токене штампа времени
    - **get_timestamp_token_request_for_signature** - Получить запрос на получения токена штампа времени для подписи

- **crl** - Плагин для работы с CRL (ЗАГЛУШКА)
    - **open_crl** - Открывает CRL
    - **get_crl_info** - Получить информацию о CRL

- **open_crl_file** - Открывает CRL из файла
- **check_certificate** - Проверка статуса сертификата по CRL
- **verify_crl** - Верификация CRL

- **pfx** - Плагин для работы с файлами хранилища ключей формата PFX
  - **verify_password** - Проверить пароль хранилища ключей
  - **list_certificates** - Получить список сертификатов пользователя
  - **list_disks** - Получить список дисков
  - **change_password** - Изменить пароль хранилища ключей
  - **save_temporary_pfx** - Сохранить ключевую пару и самоподписанный сертификат во временный файл формата PFX
  - **unload_key** - Удалить загруженные ключи по идентификатору
  - **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время
  - **list_all_certificates** - Получить список всех сертификатов пользователя
  - **save_pfx** - Сохранить ключевую пару или существующий ключ и новые сертификаты в новый файл формата PFX

- **truststore-jks** - Плагин для работы с хранилищами доверенных сертификатов формата JKS (ЗАГЛУШКА)
  - **open_truststore** - Открывает хранилище доверенных сертификатов 'truststore.jks' в домашней директории пользователя

- **pkcs10** - Плагин для генерации ключевой пары и формирования запроса на сертификат формата PKCS#10
  - **create_pkcs10_from_key** - Формировать запрос на сертификат формата PKCS#10 из существующего ключа

```
CAPIWS.callFunction({
    plugin    :"pkcs10",
    name      :"create_pkcs10_from_key",
    arguments :[
      //Идентификато ключа
      id,
      //Имя субъекта в формате X.500 или '' если нужно получить имя из
сертификата по идентификатору ключа
      replacement_x500_name,
      //OIDы политик применения сертификатов разделенных запятой
      cp_list
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **generate_keypair** - Сгенерировать ключевую пару

```
CAPIWS.callFunction({
    plugin    :"pkcs10",
    name      :"generate_keypair",
    arguments :[
      //Название алгоритма
      alg_name,
      //Случайные данные для инициализации генератора случайных чисел
      seed
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **create_pkcs10** - Формировать запрос на сертификат формата PKCS#10

```
CAPIWS.callFunction({
    plugin    :"pkcs10",
    name      :"create_pkcs10",
    arguments :[
      //Идентификатор ключевой пары
      id,
      //Имя субъекта в формате X.500
      subject_x500_name,
      //OIDы политик применения сертификатов разделенных запятой
      cp_list
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_pkcs10_info** - Получить информацию о запросе PKCS#10

```
CAPIWS.callFunction({
    plugin    :"pkcs10",
    name      :"get_pkcs10_info",
    arguments :[
      //Сертификат в кодировке BASE64 или PEM
      pkcs10
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **x509** - Плагин для работы с сертификатами X.509

- **verify_certificate** - Верификация подписи сертификата субъектка сертификатом издателя

```
CAPIWS.callFunction({
    plugin    :"x509",
    name      :"verify_certificate",
    arguments :[
      //Сертификат субъекта в кодировке BASE64
      subject_certificate_64,
      //Сертификат издателя в кодировке BASE64
      issuer_certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_certificate_info** - Получить информацию о сертификате

```
CAPIWS.callFunction({
    plugin    :"x509",
    name      :"get_certificate_info",
    arguments :[
      //Сертификат в кодировке BASE64
      certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_certificate_chain** - Получить цепочку сертификатов в кодировке BASE64 по идентификатору ключа

```
CAPIWS.callFunction({
    plugin    :"x509",
    name      :"get_certificate_chain",
    arguments :[
      //Идентификатор ключа
      certId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **truststore** - Плагин для работы с хранилищами доверенных сертификатов (ЗАГЛУШКА)

- **list_truststore** - Получить список доверенных сертификатов (ЗАГЛУШКА)

```
CAPIWS.callFunction({
    plugin    :"truststore",
    name      :"list_truststore",
    arguments :[
      //
      tsid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **ftjc** - Плагин для работы с USB-токеном FT Javacard (ЗАГЛУШКА)
  - **set_name** - Установить название USB-токену

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"set_name",
    arguments :[
      //Идентификатор ключа
      tokenId,
      //Название (не более 80 байтов)
      name
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

  - **unload_key** - Удалить загруженные ключи по идентификатору

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"unload_key",
    arguments :[
      //Идентификатор ключа
      tokenId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"load_key",
    arguments :[
      //Идентификатор токена
      card_uid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **store_certificates** - Сохранить сертификаты в USB-токен

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"store_certificates",
    arguments :[
      //Идентификатор ключа
      tokenId,
      //Сертификат субъекта в кодировке BASE64
      subject_certificate_64,
      //Сертификат Центра Регистрации в кодировке BASE64
      ca_certificate_64,
      //Корневой сертификат в кодировке BASE64
      root_certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_tokens** - Получить список токенов

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"list_tokens"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_all_keys** - Получить список всех ключей из подключенных токенов

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"list_all_keys",
    arguments :[
      //Исключить токены с идентификатором (разделенных запятой)
      except_cards
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **install_applet** - Загрузить апплет в USB-токен

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"install_applet",
    arguments :[
      //Идентификатор токена
      card_uid,
      //Подписанный апплет в кодировке BASE64
      applet_64,
      //Подпись (в формате HEX) апплета
      signature_hex
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_user_data** - Извлечь пользовательские данные

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"get_user_data",
    arguments :[
      //Идентификатор ключа
      tokenId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **change_pin** - Изменить ПИН код токена

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"change_pin",
    arguments :[
      //Идентификатор ключа
      tokenId,
      //Тип ПИН кода: 0 - Инициализации, 1 - Пользовательский, 2 - Сброс
      pinType
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_pin** - Проверить ПИН код токена

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"verify_pin",
    arguments :[
      //Идентификатор ключа
      tokenId,
      //Тип ПИН кода: 0 - Инициализации, 1 - Пользовательский, 2 - Сброс
      pinType
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_random_data** - Создать случайные данные в USB-токене

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"get_random_data",
    arguments :[
      //Идентификатор ключа
      tokenId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **set_user_data** - Записать или Удалить пользовательские данные

```
CAPIWS.callFunction({
    plugin    :"ftjc",
    name      :"set_user_data",
    arguments :[
      //Идентификатор ключа
      tokenId,
      //Данные в кодировке BASE64 или '' для удаления заранее сохраненых данных
      data_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **ytks** - Плагин для работы с файлами хранилища ключей формата YTKS
  - **verify_password** - Проверить пароль хранилища ключей

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"verify_password",
    arguments :[
      //Идентификатор ключа
      ytksId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

  - **list_certificates** - Получить список сертификатов пользователя

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"list_certificates",
    arguments :[
      //Диск
      disk
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_disks** - Получить список дисков

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"list_disks"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **change_password** - Изменить пароль хранилища ключей

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"change_password",
    arguments :[
      //Идентификатор ключа
      ytksId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **unload_key** - Удалить загруженные ключи по идентификатору

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"unload_key",
    arguments :[
      //Идентификатор ключа
      ytksId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"load_key",
    arguments :[
      //Диск
      disk,
      //Путь (должна быть пустой или 'DSKEYS')
      path,
      //Имя файла без расширения
      name,
      //Алиас ключа
      alias,
      //Серийный номер сертификата (HEX)
      serialNumber
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_all_certificates** - Получить список всех сертификатов пользователя

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"list_all_certificates"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **save_ytks** - Сохранить ключевую пару или существующий ключ и новые сертификаты в новый файл формата YTKS

```
CAPIWS.callFunction({
    plugin    :"ytks",
    name      :"save_ytks",
    arguments :[
      //Диск
      disk,
      //Путь (должна быть пустой или 'DSKEYS')
      path,
      //Имя файла без расширения
      name,
      //Алиас ключа
      alias,
      //Идентификатор новой ключевой пары или существующего хранилища ключей (для обновления сертификатов)
      id,
      //Пароль для нового ключа
      new_key_password,
      //Сертификат субъекта в кодировке BASE64
      subject_certificate_64,
      //Сертификат Центра Регистрации в кодировке BASE64
      ca_certificate_64,
      //Корневой сертификат в кодировке BASE64
      root_certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **cipher** - Плагин для шифрования и дешифрования документа по алг.шифрования ГОСТ-28147, алг.обмена ключа ECDH-SHA256 в режиме P2P
  - **decrypt_document** - Дешифровать зашифрованный документ

```
CAPIWS.callFunction({
    plugin    :"cipher",
    name      :"decrypt_document",
    arguments :[
      //Зашифрованные данные в кодировке BASE64 (будут предваритьльно декодированы)
      encrypted_64,
      //Идентификатор ключа для генерации общего секретного ключа дешифрования (полученный из фукнции других плагинов)
      id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **encrypt_document** - Создать зашифрованный документ

```
CAPIWS.callFunction({
    plugin    :"cipher",
    name      :"encrypt_document",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы, зашифрованы
и вложены в документ)
      data_64,
      //Идентификатор ключа для генерации общего секретного ключа шифрования
(полученный из фукнции других плагинов)
      id,
      //Сертификат получателя
      recipient_certificate_64,
      //Сертификат ЦРК получателя
      ca_certificate_64,
      //S-Box алг.шифрования ГОСТ-28147 (GOST28147_E_A, GOST28147_E_B,
GOST28147_E_C, GOST28147_E_D, GOST28147_D_A) (может быть пустым)
      engine,
      //Режим поточного шифрования (CFB, OFB, SIC, GOFB, GCFB) (может быть
пустым)
      mode
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **idcard** - Плагин для работы с ID-card E-IMZO
  - **verify_password** - Проверить пароль хранилища ключей (заглушка)

```
CAPIWS.callFunction({
    plugin    :"idcard",
    name      :"verify_password",
    arguments :[
      //Идентификатор ключа
      pfxId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **personalize** - Персонализировать ID-карту записав новые сертификаты и установив PIN-код

```
CAPIWS.callFunction({
    plugin    :"idcard",
    name      :"personalize",
    arguments :[
      //PIN-код
      pincode,
      //Сертификат субъекта в кодировке BASE64
      subject_certificate_64,
      //Сертификат Центра Регистрации в кодировке BASE64
      ca_certificate_64,
      //Корневой сертификат в кодировке BASE64
      root_certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_readers** - Получить список считывателей

```
CAPIWS.callFunction({
    plugin    :"idcard",
    name      :"list_readers"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время (заглушка)

```
CAPIWS.callFunction({
    plugin    :"idcard",
    name      :"load_key",
    arguments :[
      //Диск
      disk,
      //Путь
      path,
      //Имя файла без расширения
      name,
      //Алиас ключа
      alias
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_encrypted_signed_cplc** - Получить зашифрованный и подписанный заводской номер USB-токена

```
CAPIWS.callFunction({
    plugin    :"idcard",
    name      :"get_encrypted_signed_cplc",
    arguments :[
      //Значение NONCE
      nonce
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_all_certificates** - Получить список всех сертификатов пользователя (заглушка)

```
CAPIWS.callFunction({
    plugin    :"idcard",
    name      :"list_all_certificates"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **fileio** - Плагин для работы с файлами
  - **load_file** - Загруить файл

```
CAPIWS.callFunction({
    plugin    :"fileio",
    name      :"load_file",
    arguments :[
      //Путь к файлу
      path
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **write_file** - Записать содержимое zip файла на диск

```
CAPIWS.callFunction({
    plugin    :"fileio",
    name      :"write_file",
    arguments :[
      //Диск
      disk,
      //Zip файл в кодировке BASE64
      zip_64,
      //Подпись (в формате HEX) Zip файла
      signature_hex
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **tunnel** - Плагин для установки зашифрованное соединения с сервером по алг.шифрования ГОСТ-28147
  - **create_tunnel** - Создать зашифрованного соединения с сервером и вернуть TCP-порт для приема/передачи данных

```
CAPIWS.callFunction({
    plugin    :"tunnel",
    name      :"create_tunnel",
    arguments :[
      //Идентификатор адреса назначения (настраивается на сервере)
      dst_id,
      //Идентификатор ключа для генерации общего секретного ключа шифрования
(полученный из фукнции других плагинов)
      id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **pki** - Плагин для взаимодействия с ИОК

- **enroll_pfx_step2** - Шаг №2 для получения ключа PFX

```
CAPIWS.callFunction({
    plugin    :"pki",
    name      :"enroll_pfx_step2",
    arguments :[
      //Идентификатор процесса GUID (полученный из Шага №1)
      guid,
      //Сертификат субъекта в кодировке BASE64
      subject_certificate_64,
      //Сертификат Центра Регистрации в кодировке BASE64
      ca_certificate_64,
      //Корневой сертификат в кодировке BASE64
      root_certificate_64,
      //Данные в кодировке BASE64 (будут предваритьльно декодированы, подписаны и
вложены в документ)
      data_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **enroll_pfx_step1** - Шаг №1 для получения ключа PFX

```
CAPIWS.callFunction({
    plugin    :"pki",
    name      :"enroll_pfx_step1",
    arguments :[
      //Идентификатор процесса GUID
      guid,
      //Название алгоритма
      alg_name,
      //Случайные данные для инициализации генератора случайных чисел
      seed,
      //Имя субъекта в формате X.500
      subject_x500_name,
      //OIDы политик применения сертификатов разделенных запятой
      cp_list,
      //Имя файла без расширения (если тип носителя ключа - файл)
      file_name,
      //Данные в кодировке BASE64 (будут предваритьльно декодированы, подписаны и
вложены в документ)
      data_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **pkcs7** - Плагин для работы с форматом PKCS#7/CMS

- **create_pkcs7** - Создать PKCS#7/CMS документ подписав ключем задаваемым идентификатором

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"create_pkcs7",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы, подписаны и вложены в документ)
      data_64,
      //Идентификатор ключа подписывающего лица (полученный из фукнции других плагинов)
      id,
      //Возможные значения: 'yes' - будет создан PKCS#7/CMS документ без вложения исходных данных, 'no' или '' - будет создан PKCS#7/CMS документ с вложением исходных данных
      detached
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **attach_timestamp_token_pkcs7** - Прикрепить токен штампа времени к документу PKCS#7/CMS (ЗАГЛУШКА)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"attach_timestamp_token_pkcs7",
    arguments :[
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Серийный номер сертификата подписавшего документ, на подпись которого будет прикреплен токен штампа времени
      signer_serial_number,
      //Токен штампа времени (полученный из фукнции других плагинов)
      timestamp_token_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_pkcs7_attached** - Верифицировать документ PKCS#7/CMS, который содержит исходный документ (ЗАГЛУШКА)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"verify_pkcs7_attached",
    arguments :[
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции
других плагинов), для верификация сертификатов
      tsid,
      //Идентификатор ключа запросителя онлайн статуса сертификата (полученный из
фукнции других плагинов), для проверки статуса сертификата в режиме онлайн по
протоколу OCSP. Применяется для подписания OCSP запросов, если этого требует OCSP
сервис, иначе может быть пустым
      rid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_pkcs7_detached** - Верифицировать документ PKCS#7/CMS, который не содержит исходный документ (ЗАГЛУШКА)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"verify_pkcs7_detached",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы)
      data_64,
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции
других плагинов), для вырификация сертификатов
      tsid,
      //Идентификатор ключа запросителя онлайн статуса сертификата (полученный из
фукнции других плагинов), для проверки статуса сертификата в режиме онлайн по
протоколу OCSP. Применяется для подписания OCSP запросов, если этого требует OCSP
сервис, иначе может быть пустым
      rid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_pkcs7_detached_crl** - Верифицировать документ PKCS#7/CMS, который не содержит исходный документ (ЗАГЛУШКА)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"verify_pkcs7_detached_crl",
    arguments :[
      //Данные в кодировке BASE64 (будут предварительно декодированы)
      data_64,
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции других плагинов), для верификация сертификатов
      tsid,
      //Идентификатор CRL (полученный из фукнции других плагинов), для проверки статуса сертификата в режиме оффлайн
      cid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **append_pkcs7_attached** - Добавить подпись в существующий документ PKCS#7/CMS (DEPRECATED)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"append_pkcs7_attached",
    arguments :[
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор ключа подписывающего лица (полученный из фукнции других плагинов)
      id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_pkcs7_detached_info** - Получить полную информацию о документе PKCS#7/CMS, который не содержит исходный документ

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"get_pkcs7_detached_info",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы)
      data_64,
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции
других плагинов), если требуется верификация сертификатов, иначе может быть
пустым
      tsid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_pkcs7_attached_crl** - Верифицировать документ PKCS#7/CMS, который содержит исходный документ (ЗАГЛУШКА)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"verify_pkcs7_attached_crl",
    arguments :[
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции
других плагинов), для верификация сертификатов
      tsid,
      //Идентификатор CRL (полученный из фукнции других плагинов), для проверки
статуса сертификата в режиме оффлайн
      cid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **append_pkcs7_detached** - Добавить подпись в существующий документ PKCS#7/CMS, который не содержит исходного документа (DEPRECATED)

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"append_pkcs7_detached",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы для подписания)
      data_64,
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор ключа подписывающего лица (полученный из фукнции других плагинов)
      id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_pkcs7_attached_info** - Получить полную информацию о документе PKCS#7/CMS, который содержит исходный документ

```
CAPIWS.callFunction({
    plugin    :"pkcs7",
    name      :"get_pkcs7_attached_info",
    arguments :[
      //Ранее созданный документ PKCS#7/CMS в кодировке BASE64
      pkcs7_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции других плагинов), если требуется верификация сертификатов, иначе может быть пустым
      tsid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **cryptoauth** - Плагин для выполнения низкоуровневых криптографических преобразований (ЗАГЛУШКА)

- **get_signature** - Подписать данные ключем задаваемым идентификатором

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"get_signature",
    arguments :[
      //Данные в кодировке BASE64 (будут предварительно декодированы, перед хешированием)
      data_64,
      //Идентификатор ключа подписывающего лица (полученный из фукнции других плагинов)
      id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_digest_hex** - Вычислить хеш (в формате HEX) данных по алгоритму OZDST-1106-2009-2-A

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"get_digest_hex",
    arguments :[
      //Данные в кодировке BASE64 (будут предварительно декодированы, перед хешированием)
      data_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_digest_hex_signature_with_id** - Верифицировать подпись хеша (в формате HEX) ключем задаваемым идентификатором

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"verify_digest_hex_signature_with_id",
    arguments :[
      //Хеш (в формате HEX) полученный ранее
      digest_hex,
      //Подпись (в формате HEX) полученный другой функцией
      signature_hex,
      //Сертификат определяемый идентификатором ключа подписавшего лица (полученный из фукнции других плагинов)
      certId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_signature_with_certificate** - Верифицировать подпись данных сертификатом

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"verify_signature_with_certificate",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы, перед хешированием)
      data_64,
      //Подпись (в формате HEX) полученный другой функцией
      signature_hex,
      //Сертификат в кодировке BASE64 подписавшего лица (полученный из фукнции других плагинов)
      certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_digest_hex_signature** - Подписать хеш (в формате HEX) ключем задаваемым идентификатором

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"get_digest_hex_signature",
    arguments :[
      //Хеш (в формате HEX) полученный ранее
      digest_hex,
      //Идентификатор ключа подписывающего лица (полученный из фукнции других плагинов)
      id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_signature_with_id** - Верифицировать подпись данных ключем задаваемым идентификатором

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"verify_signature_with_id",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы, перед хешированием)
      data_64,
      //Подпись (в формате HEX) полученный другой функцией
      signature_hex,
      //Сертификат определяемый идентификатором ключа подписавшего лица (полученный из фукнции других плагинов)
      certId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_digest_hex_signature_with_certificate** - Верифицировать подпись хеша (в формате HEX) сертификатом

```
CAPIWS.callFunction({
    plugin    :"cryptoauth",
    name      :"verify_digest_hex_signature_with_certificate",
    arguments :[
      //Хеш (в формате HEX) полученный ранее
      digest_hex,
      //Подпись (в формате HEX) полученный другой функцией
      signature_hex,
      //Сертификат в кодировке BASE64 подписавшего лица (полученный из фукнции других плагинов)
      certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **certkey** - Плагин для работы с электронными ключами и сертификатами (ЗАГЛУШКА)
  - **unload_key** - Удалить загруженные ключи по идентификатору

```
CAPIWS.callFunction({
    plugin    :"certkey",
    name      :"unload_key",
    arguments :[
      //Идентификатор ключа
      keyId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

  - **list_certificates** - Получить список сертификатов пользователя

```
CAPIWS.callFunction({
    plugin    :"certkey",
    name      :"list_certificates",
    arguments :[
      //Диск
      disk
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_disks** - Получить список дисков

```
CAPIWS.callFunction({
    plugin    :"certkey",
    name      :"list_disks"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время

```
CAPIWS.callFunction({
    plugin    :"certkey",
    name      :"load_key",
    arguments :[
      //Диск
      disk,
      //Путь (должна быть пустой или 'DSKEYS')
      path,
      //Имя файла без расширения
      name,
      //Серийный номер сертификата (HEX)
      serialNumber
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_all_certificates** - Получить список всех сертификатов пользователя

```
CAPIWS.callFunction({
    plugin    :"certkey",
    name      :"list_all_certificates"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **tsaclient** - Плагин для работы с токенами штампов времени

- **get_timestamp_token_for_signature** - Получить токен штампа времени на подпись от службы штампов времени по веб-ссылке

```
CAPIWS.callFunction({
    plugin    :"tsaclient",
    name      :"get_timestamp_token_for_signature",
    arguments :[
      //Подпись в формате HEX
      signature_hex,
      //Веб-ссылка службы штампов времени
      tsaUrl
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_timestamp_token_request_for_data** - Получить запрос на получения токена штампа времени для данных

```
CAPIWS.callFunction({
    plugin    :"tsaclient",
    name      :"get_timestamp_token_request_for_data",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы)
      data_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_timestamp_token_for_data** - Получить токен штампа времени на данные от службы штампов времени по веб-ссылке

```
CAPIWS.callFunction({
    plugin    :"tsaclient",
    name      :"get_timestamp_token_for_data",
    arguments :[
      //Данные в кодировке BASE64 (будут предваритьльно декодированы)
      data_64,
      //Веб-ссылка службы штампов времени
      tsaUrl
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_timestamp_token_info** - Получить информацию о токене штампа времени

```
CAPIWS.callFunction({
    plugin    :"tsaclient",
    name      :"get_timestamp_token_info",
    arguments :[
      //Токен штампа времени в кодировке BASE64
      timestamp_token_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_timestamp_token_request_for_signature** - Получить запрос на получения токена штампа времени для подписи

```
CAPIWS.callFunction({
    plugin    :"tsaclient",
    name      :"get_timestamp_token_request_for_signature",
    arguments :[
      //Подпись в формате HEX
      signature_hex
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **crl** - Плагин для работы с CRL (ЗАГЛУШКА)
  - **open_crl** - Открывает CRL

```
CAPIWS.callFunction({
    plugin    :"crl",
    name      :"open_crl",
    arguments :[
      //CRL в кодировке BASE64
      crl_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **get_crl_info** - Получить информацию о CRL

```
CAPIWS.callFunction({
    plugin    :"crl",
    name      :"get_crl_info",
    arguments :[
      //Идентификатор CRL (полученный из фукнции других плагинов)
      cid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **open_crl_file** - Открывает CRL из файла

```
CAPIWS.callFunction({
    plugin    :"crl",
    name      :"open_crl_file",
    arguments :[
      //Идентификатор файла (полученный из фукнции других плагинов)
      file_id
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **check_certificate** - Проверка статуса сертификата по CRL

```
CAPIWS.callFunction({
    plugin    :"crl",
    name      :"check_certificate",
    arguments :[
      //Идентификатор CRL (полученный из фукнции других плагинов)
      cid,
      //Проверяемый сертификат в кодировке BASE64
      subject_certificate_64,
      //Дата проверки в формате 'yyyy.MM.dd HH:mm:ss'
      check_date
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **verify_crl** - Верификация CRL

```
CAPIWS.callFunction({
    plugin    :"crl",
    name      :"verify_crl",
    arguments :[
      //Идентификатор CRL (полученный из фукнции других плагинов)
      cid,
      //Сертификат издателя CRL в кодировке BASE64
      crl_issuer_certificate_64,
      //Идентификатор хранилища доверенных сертификатов (полученный из фукнции
других плагинов), для верификации сертификата издателя CRL
      tsid
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **pfx** - Плагин для работы с файлами хранилища ключей формата PFX
  - **verify_password** - Проверить пароль хранилища ключей

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"verify_password",
    arguments :[
      //Идентификатор ключа
      pfxId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

  - **list_certificates** - Получить список сертификатов пользователя

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"list_certificates",
    arguments :[
      //Диск
      disk
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_disks** - Получить список дисков

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"list_disks"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **change_password** - Изменить пароль хранилища ключей

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"change_password",
    arguments :[
      //Идентификатор ключа
      pfxId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **save_temporary_pfx** - Сохранить ключевую пару и самоподписанный сертификат во временный файл формата PFX

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"save_temporary_pfx",
    arguments :[
      //Диск
      disk,
      //Путь (должна быть пустой или 'DSKEYS')
      path,
      //Имя файла без расширения
      name,
      //Алиас ключа
      alias,
      //Идентификатор новой ключевой пары
      id,
      //Пароль для временного ключа
      password,
      //Имя субъекта в формате X.500
      subject_x500_name
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **unload_key** - Удалить загруженные ключи по идентификатору

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"unload_key",
    arguments :[
      //Идентификатор ключа
      pfxId
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **load_key** - Загрузить ключ и получить идентификатор ключа. Ключ будет доступен определенное время

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"load_key",
    arguments :[
      //Диск
      disk,
      //Путь (должна быть пустой или 'DSKEYS')
      path,
      //Имя файла без расширения
      name,
      //Алиас ключа
      alias
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **list_all_certificates** - Получить список всех сертификатов пользователя

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"list_all_certificates"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **save_pfx** - Сохранить ключевую пару или существующий ключ и новые сертификаты в новый файл формата PFX

```
CAPIWS.callFunction({
    plugin    :"pfx",
    name      :"save_pfx",
    arguments :[
      //Диск
      disk,
      //Путь (должна быть пустой или 'DSKEYS')
      path,
      //Имя файла без расширения
      name,
      //Алиас ключа
      alias,
      //Идентификатор новой ключевой пары или существующего хранилища ключей (для обновления сертификатов)
      id,
      //Пароль для нового ключа
      new_key_password,
      //Сертификат субъекта в кодировке BASE64
      subject_certificate_64,
      //Сертификат Центра Регистрации в кодировке BASE64
      ca_certificate_64,
      //Корневой сертификат в кодировке BASE64
      root_certificate_64
    ]
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```

- **truststore-jks** - Плагин для работы с хранилищами доверенных сертификатов формата JKS (ЗАГЛУШКА)
  - **open_truststore** - Открывает хранилище доверенных сертификатов 'truststore.jks' в домашней директории пользователя

```
CAPIWS.callFunction({
    plugin    :"truststore-jks",
    name      :"open_truststore"
  },
  function(event, data){
    console.log(data);
  },
  function(error){
    window.alert(error);
  }
);
```