WebSocket клиент и сервер для 1С. Внешний компонент (Документация)

Выпуск 1

best-tech

февр. 05, 2020

Содержание:

Внешний компонент для 1С: Предприятие выполнен по технолонии NativeAPI.

поставка решения https://infostart.ru/public/937068/

Позволяет обмениваться по протоколу WebSocket, имеет возможность выступать в роли Клиента и Сервера (ssl/nossl)

Глава 1

Подключение компонента

1.1 &НаСервере

`ПодключитьВнешнююКомпоненту()`
Синоним: **AttachAddIn()**

Подключает компонент, выполненный по технологии Native API. Компонент может храниться в информационной базе или макете конфигурации в виде двоичных данных или в ZIP-архиве. Для режимов запуска «Тонкий клиент» и «Веб-клиент», компонент должен быть предварительно установлен методом `УстановитьВнешнююКомпоненту()`.

Пример использования:

```
ПутьКомпоненты = "ОбщийМакет.МакетКомпоненты";

Если НЕ ПодключитьВнешнююКомпоненту(ПутьКомпоненты,"WebSocket", ТипВнешнейКомпоненты.Native) Тогда
    ВызватьИсключение "Ошибка подключения внешнего компонента";
КонецЕсли;
```

1.2 &НаКлиенте

`УстановитьВнешнююКомпоненту()`
Синоним: **InstallAddIn()**

Доставляет объект внешнего компонента с сервера на клиент, после чего он становится доступен для метода `ПодключитьВнешнююКомпоненту()`.

Примечание:

Метод работает только с компонентами, хранящимися в архиве.

Если для конфигурации свойство РежимИспользованияМодальности установлено в НеИспользовать, следует использовать метод `НачатьУстановкуВнешнейКомпоненты()`.

Пример использования:

```
ПутьКомпоненты = "ОбщийМакет.МакетКомпоненты";

УстановитьВнешнююКомпоненту(ПутьКомпоненты);

Если НЕ ПодключитьВнешнююКомпоненту(ПутьКомпоненты,"WebSocket", ТипВнешнейКомпоненты.Native) Тогда
    ВызватьИсключение "Ошибка подключения внешнего компонента";
КонецЕсли;
```

1.3 MANIFEST.XML

Пример файла:

```
<?xml version="1.0" encoding="UTF-8"?>
<bundle xmlns="http://v8.1c.ru/8.2/addin/bundle" name="websocket">
    <component os="Windows" path="1c-websocket_Win_32.dll" type="native" arch="i386"/>
    <component os="Windows" path="1c-websocket_Win_64.dll" type="native" arch="x86_64"/>
</bundle>
```

Глава 2

Клиент WebSocket

2.1 Инициализация

`Новый("AddIn.WebSocket.Client");`

Синоним: **New()**

Инициализация внешнего компонента (Клмент). Полученный таким образом объект используется для дальнейшего взаимодействия.

Пример использования:

```
1 Клиент = Новый("AddIn.WebSocket.Client");
```

2.2 Подключение к серверу

`Подключиться(АдресСервера)`

Синоним: **Connect()**

Выполняет подключение к серверу, блокируется вызов до решения о подключение. Состояние подключения можно определить свойством `Статус`

После подключение прием и отправка сообщений выполняется в фоновом режиме соответствующими методами `Получить` и `Отправить`.

Параметры:

АдресСервера - **Строка** - Полный адрес сервера с протоколом и портом, например `wss://echo.websocket.org`

Пример использования:

```
1 АдресСервера = "wss://echo.websocket.org";
2
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    Попытка
            Клиент.Подключиться(АдресСервера);
    Исключение

            Описание = ОписаниеОшибки();
            ТекстОшибки =  Клиент.ОписаниеОшибки();

            ТекстОписания = Описание + ": " + ТекстОшибки;

            ВызватьИсключение ТекстОписания;

    КонецПопытки;
```

2.3 Отключение от сервера

`Отключиться()`

Синоним: Disconnect()

Выполняет отключение от сервера.

Пример использования:

```
Клиент.Отключиться();
```

2.4 Отправка сообщения

`Отправить(ТелоСообщения)`

Синоним: **Send()**

Добавляет сообщение для фоновой отправки клиенту

Параметры:

ТелоСообщения - **Строка** - Тело отправляемого сообщения

Пример использования:

```
ТелоСообщения = "Hello World 1C";

Сервер.Отправить(ТелоСообщения);
```

2.5 Получение сообщения

`Получить(Таймаут, Данные)`

Синоним: **Receive()**

Получает принятое сообщение в фоновом режиме, если сообщений нет ждет `Таймаут` миллисекунд, или ждет до появления сообщения, если `Таймаут=0`

Возвращаемое значение:

Тип: **Булево**

Описание: Признак принятия сообщения. Истина, если сообщение принято

Параметры:

Таймаут - **Число** - Количество миллисекунд, на которое заблокировать вызов и ожидать сообщения. Если 0 - ждать бесконечно.

Данные - **Строка** - Полученные данные

Пример использования:

```
Таймаут = 0;
Данные = "";

// Постоянный цикл принятие сообщений
Пока Клиент.Принять(Таймаут, Данные) Цикл

    Сообщить("Принят пакет данных:");
    Сообщить(Данные);

КонецЦикла;

Таймаут = 3000;

// Ждем 3 секунды, если не принято сообщений исполнение кода продолжается
Пока Клиент.Принять(Таймаут, Данные) Цикл

    Сообщить("Принят пакет данных:");
    Сообщить(Данные);

КонецЦикла;

Сообщить("Закончен прием сообщений");
```

2.6 Статус

`Статус`

Синоним: **Status**

Возвращаемое значение:

Тип: **Число**

Описание: Текущий статус подключения

`-1` - Ошибка подключения

`0` - Не подключено

`1` - Подключено

Пример использования:

```
Если Клиент.Статус = 1 Тогда
    // Можно выполнять отправку данных
КонецЕсли;
```

2.7 Сервер

`Сервер`

Синоним: **Server**

Возвращаемое значение:

Тип: **Строка**

Описание: Хранит адрес текущего соединения

Пример использования:

```
Если Клиент.Сервер = 1 Тогда
    Сообщить("Сейчас подключено к серверу: " + Клиент.Сервер);
КонецЕсли;
```

Глава 3

Сервер WebSocket

3.1 Инициализация

`Новый("AddIn.WebSocket.Server");`

Синоним: **New()**

Инициализация внешнего компонента (Сервер).

Полученный таким образом объект используется для дальнейшего взаимодействия.

Пример использования:

```
1 Сервер = Новый("AddIn.WebSocket.Server");
```

3.2 Прослушивание порта

`Запустить(Интерфейс, Порт, Сертификат="", Пароль="")`

Синоним: **Run()**

Захватывает порт для прослушивания и принимает в асинхронном режиме соединения и сообщения. Для получения тела сообщения используется метод `Получить()`

Возвращаемое значение:

Тип: **Булево**

Описание: Удалось ли захватить порт и начать прослушивание

Параметры:

Интерфейс - **Строка** - Интерфейс для прослушивания, например `"0.0.0.0."`

Порт - **Число** - Числовой порт для прослушивания, например `8080`

Сертификат - **Строка** - (Необязательный) Путь (ANSI) к сертификату для организации SSL доступа (`-----BEGIN RSA PRIVATE KEY-----` + `-----BEGIN CERTIFICATE-----`)

Пароль - **Строка** - (Необязательный) Пароль для сертификата

Пример использования:

```
Попытка

    Сервер.Запустить("127.0.0.1", 8098, "c:/misc/ssl/cert.pem", "c:/misc/ssl/key.pem", "1234");

Исключение

    Описание = ОписаниеОшибки();

    ТекстОшибки =  Сервер.ОписаниеОшибки();

    ТекстОписания = Описание + ": " + ТекстОшибки;

    ВызватьИсключение ТекстОписания;

КонецПопытки;
```

3.3 Завершение прослушивания

`Остановить()`

Синоним: **Stop()**

Завершает фоновую обработку соединений и освобождает порт. Исключений не вызывает

Пример использования:

```
Сервер.Остановить();
```

3.4 Отправка сообщения

`Отправить(ИДКлиента, ТелоСообщения)`

Синоним: **Send()**

Добавляет сообщение для фоновой отправки клиенту

Параметры:

ИДКлиента - **Число** - Порядковый номер соединения. Можно посмотреть в свойстве `Соединения` или при обработке входящий сообщений методом `Принять()`

ТелоСообщения - **Строка** - Тело отправляемого сообщения

Пример использования:

```
ТелоСообщения = "Hello World 1C";

ИДКлиента = 1;

Сервер.Отправить(ИДКлиента, ТелоСообщения);
```

3.5 Получение сообщения

`Принять(Таймаут, ИДКлиента, Данные)`

Синоним: **Receive()**

Получает принятое сообщение в фоновом режиме, если сообщений нет ждет `Таймаут` миллисекунд, или ждет до появления сообщения, если `Таймаут=0`

Возвращаемое значение:

Тип: **Булево**

Описание: Признак принятия сообщения. Истина, если сообщение принято

Параметры:

Таймаут - **Число** - Количество миллисекунд, на которое заблокировать вызов и ожидать сообщения. Если 0 - ждать бесконечно.

ИДКлиента - **Число** - Идентификатор соединения с клиентом

Данные - **Строка** - Полученные данные

Пример использования:

```
Таймаут = 0;
ИДКлиента = 1;
Данные = "";

// Постоянный цикл принятие сообщений
Пока Клиент.Принять(Таймаут, ИДКлиента, Данные) Цикл

    Сообщить("Принят пакет данных от " + СокрЛП(ИДКлиента));
    Сообщить(Данные);

КонецЦикла;

Таймаут = 3000;

// Ждем 3 секунды, если не принято сообщений исполнение кода продолжается
Пока Клиент.Принять(Таймаут, ИДКлиента, Данные) Цикл

    Сообщить("Принят пакет данных от " + СокрЛП(ИДКлиента));
    Сообщить(Данные);

КонецЦикла;

Сообщить("Закончен прием сообщений");
```

3.6 Отключение клиента

`Отключить(ИДКлиента)`

Синоним: **Disconnect()**

Возвращаемое значение:

Тип: **Булево**

Описание: Признак успешного отключения клиента

Отключает клиента от себя по идентификатору соединения

Параметры:

ИДКлиента - **Число** - Идентификатор соединения с клиентом

Пример использования:

```
ИДКлиента = 2;

Сервер.Отключить(ИДКлиента);
```

3.7 Сервер

`Сервер`

Синоним: **Server**

Возвращаемое значение:

Тип: **Строка**

Описание: Хранит адрес текущего интерфейса прослушивания

Пример использования:

```
Если ЗначениеЗаполнено(Клиент.Сервер) Тогда
    Сообщить("Сейчас сервер прослушивает: " + Клиент.Сервер);
КонецЕсли;
```

3.8 Порт

`Порт`

Синоним: **Port**

Возвращаемое значение:

Тип: **Число**

Описание: Хранит порт прослушивания

Пример использования:

```
Если ЗначениеЗаполнено(Сервер.Сервер) Тогда
    Сообщить("Сейчас сервер прослушивает порт: " + Сервер.Порт);
КонецЕсли;
```

3.9 Соединения

`Соединения`

Синоним: **Connections**

Возвращаемое значение:

Тип: **Строка**

Описание: Текущие активные соединения, `ИД1:АдресПодключения1|ИД2:АдресПодключения2`

Пример использования:

```
Если ЗначениеЗаполнено(Сервер.Сервер) Тогда
    Сообщить("Текущие соединения: " + Сервер.Соединения);
КонецЕсли;
```

Глава 4

Обработка ошибок (исключения)

Объекты, созданные с помощью компонента могу вызывать исключения. Получить описание исключения можно вызвав метод.

Если в функцию объекта передаются параметры не соответствующих типов, то в тексте описания ошибки будет содержаться «<bad cast>»

`ОписаниеОшибки()`

Возвращаемое значение:

Тип: **Строка**

Описание: Получает текстовое описание ошибки.

```
Попытка
    Сервер.Запустить("127.0.0.1", 8098);
Исключение
    Описание = ОписаниеОшибки();
    ТекстОшибки =  Сервер.ОписаниеОшибки();
    ТекстОписания = Описание + ": " + ТекстОшибки;
    ВызватьИсключение ТекстОписания;
КонецПопытки;
```

Глава 5

Полное тестирование

`ВыполнитьПолноеТестированиеКомпоненты()`

Процедура выполняет полное тестирование всех функций компонента. Можно использовать как сквозной пример.

Листинг:

```
Процедура ВыполнитьПолноеТестированиеКомпоненты()
    ПутьКомпоненты = "D:\websocket.dll";

    // ШАГ1. ТЕСТ КЛИЕНТА ЧЕРЕЗ ВНЕШНИЙ СЕРВИС
    АдресСервера = "wss://echo.websocket.org";

    Если НЕ ПодключитьВнешнююКомпоненту(ПутьКомпоненты, "websocket", ТипВнешнейКомпоненты.Native) Тогда
        СисИнфо = Новый СистемнаяИнформация;
        ОписаниеОшибки = НСтр("ru='Ошибка подключения компоненты ('") + СисИнфо.ТипПлатформы + "):
        |" + ОписаниеОшибки();

        ВызватьИсключение ОписаниеОшибки;
    КонецЕсли;

    Клиент = Новый("AddIn.WebSocket.Client");

    Попытка
        Клиент.Подключиться(Объект.АдресСервера);
    Исключение

        Описание = ОписаниеОшибки();
        ТекстОшибки =  Клиент.ОписаниеОшибки();

        ТекстОписания = Описание + ": " + ТекстОшибки;

        ВызватьИсключение ТекстОписания;

```

(continues on next page)

(продолжение с предыдущей страницы)

```
    КонецПопытки;


    Попытка
        Клиент.Подключиться(Объект.АдресСервера);
    Исключение
        ТекстОшибки =  Клиент.ОписаниеОшибки();

        Если найти(ТекстОшибки, "Подключение уже выполнено") = 0 Тогда

            ВызватьИсключение ТекстОписания;

        КонецЕсли;

    КонецПопытки;

    ДанныеОтправки = "ase1ЙЦУ";

    Клиент.Отправить(ДанныеОтправки);

    Данные = "";

    Принято = Клиент.Принять(0, Данные);

    Если НЕ Данные = ДанныеОтправки Тогда
        ВызватьИсключение "Приняты не те данные которые отправлены";
    КонецЕсли;

    Данные = "";
    Принято = Клиент.Принять(3000, Данные);

    Если Принято ИЛИ ЗначениеЗаполнено(Данные) Тогда
        ВызватьИсключение "Принято то что не дожно приниматься";
    КонецЕсли;

    Клиент.Отключиться();
    Клиент.Отключиться();

    Клиент = Неопределено;

    Сообщить("Тестирование Шаг 1. Успешно пройдено");

    // ШАГ 2. ТЕСТ СОБСТВЕННОГО СЕРВЕРА И СОБСТВЕННЫХ КЛИЕНТОВ

    АдресПодключения = "127.0.0.1";
    ПортПодключения = 9098;

    Сервер = Новый(Объект.ПрефиксКомпоненты  + "Server");

    Попытка
        Сервер.Запустить(АдресПодключения, ПортПодключения);
    Исключение

        Описание = ОписаниеОшибки();
        ТекстОшибки =  Сервер.ОписаниеОшибки();

```

(continues on next page)

(продолжение с предыдущей страницы)

```
84          ТекстОписания = Описание + ": " + ТекстОшибки;
85
86          ВызватьИсключение ТекстОписания;
87
88      КонецПопытки;
89
90      Клиент1 = Новый(Объект.ПрефиксКомпоненты  + "Client");
91      Клиент2 = Новый(Объект.ПрефиксКомпоненты  + "Client");
92
93      СтрокаПодключения  = СтрШаблон("ws://%1:%2", АдресПодключения, Формат(ПортПодключения, "ЧГ=0
↪"));
94
95      Попытка
96          Клиент1.Подключиться(СтрокаПодключения+"/first");
97          Клиент2.Подключиться(СтрокаПодключения+"/testadr");
98      Исключение
99
100         Сервер.Остановить();
101
102         Описание = ОписаниеОшибки();
103         ТекстОшибки =  Клиент1.ОписаниеОшибки();
104         ТекстОшибки = ТекстОшибки+ Клиент2.ОписаниеОшибки();
105
106         ТекстОписания = Описание + ": " + ТекстОшибки;
107
108         ВызватьИсключение ТекстОписания;
109
110     КонецПопытки;
111
112     ТекстОтправки = "ase1ЙЦУ";
113
114     Клиент1.Отправить("ase1ЙЦУ");
115
116     Данные = "";
117     ИДКлиента = 0;
118
119     Принято = Сервер.Принять(0, ИДКлиента, Данные);
120
121     Результат = Сервер.Отключить(ИдКлиента);
122
123     Клиент1 = Неопределено;
124     Клиент2 = Неопределено;
125     Сервер = Неопределено;
126
127 КонецПроцедуры
```

Глава 6

Примеры использования

6.1 slack bot

Пример реализации бота, который получает сообщений, отправленные пользователями в Slack Подробнее

Для реализации требуется token Slack (подробнее на просторах интернета)

При подключении бот отправляет приветственное сообщение и отображает в 1С сообщения из чата.

`SlackBot()`

Описание: Пример бота Slack с постоянным подключения и отслеживанием событий

```
1  // Запустить процедуру отправить сообщение в канал
2  Процедура SlackBot()
3
4      ПутьККомпоненте = "c:/websocket.dll";
5      ИДКанала = "C23535436TR";
6      ТокенСлак = "xora-3234234324....";
7
8      Заголовки = Новый Соответствие;
9      Заголовки.Вставить("Authorization", "Bearer " + ТокенСлак);
10
11     Запрос = Новый HTTPЗапрос("api/rtm.connect", Заголовки);
12
13     Соединение = Новый HTTPСоединение("slack.com",,,,,, Новый ЗащищенноеСоединениеOpenSSL);
14
15     Ответ = Соединение.Получить(Запрос);
16
17     Если НЕ Ответ.КодСостояния = 200 Тогда
18         ВызватьИсключение "Не верный ответ";
19     КонецЕсли;
20
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    Данные =  ПолучитьЗначениеИзОтветаJSON(Ответ.ПолучитьТелоКакСтроку());

    урл = Данные.Получить("url");

    Если НЕ ЗначениеЗаполнено(урл) Тогда
        ВызватьИсключение "Нет адреса подключения";
    КонецЕсли;

    #Если Не Сервер Тогда
    //УстановитьВнешнююКомпоненту(ПутьККомпоненте);
    #КонецЕсли

    Если НЕ ПодключитьВнешнююКомпоненту(ПутьККомпоненте, "WebSocket", ТипВнешнейКомпоненты.
↪Native) Тогда
        СисИнфо = Новый СистемнаяИнформация;
        ОписаниеОшибки = НСтр("ru='Ошибка подключения компоненты ('") + СисИнфо.ТипПлатформы + "):
        |" + ОписаниеОшибки();

        ВызватьИсключение ОписаниеОшибки;
    КонецЕсли;

    Клиент = Новый("AddIn.WebSocket.Client");

    ТекстСообщения = ПолучитьСтрокуJSON(Новый Структура("type, channel, text", "message",␣
↪ИДКанала, "Listen for 1C Enterprise"));

    Попытка
        Клиент.Подключиться(урл);
    Исключение

        Описание = ОписаниеОшибки();
        ТекстОшибки =  Клиент.ОписаниеОшибки();

        ТекстОписания = Описание + ": " + ТекстОшибки;

        ВызватьИсключение ТекстОписания;

    КонецПопытки;

    Данные = "";

    ГотовПринимать = Ложь;

    Пока Клиент.Принять(0, Данные) Цикл

        Значение = ПолучитьЗначениеИзОтветаJSON(Данные);

        Текст = Значение.Получить("text");

        Если Значение.Получить("type") = "hello" Тогда
            Клиент.Отправить(ТекстСообщения);
            ГотовПринимать = Истина;
        КонецЕсли;

        Если Текст = Неопределено ИЛИ НЕ Значение.Получить("reply_to") = Неопределено ИЛИ НЕ␣
↪ГотовПринимать Тогда
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            Продолжить;
        КонецЕсли;

        Если НРег(Текст) = НРег("go away!") ИЛИ НРег(Текст) = НРег("11") Тогда

            ТекстСообщения = ПолучитьСтрокуJSON(Новый Структура("type, channel, text", "message",␣
↪ИДКанала, "ok. bye-bye )))"));
            Клиент.Отправить(ТекстСообщения);

            Сообщить("Меня отключили");
            Прервать;

        КонецЕсли;

        Сообщить(Текст);

        ТекстСообщения = ПолучитьСтрокуJSON(Новый Структура("type, channel, text", "message",␣
↪ИДКанала, "Получил: "+ Текст));
        Клиент.Отправить(ТекстСообщения);

    КонецЦикла;

    Клиент.Отключиться();

    //Клиент = Неопределено;


КонецПроцедуры


Функция ПолучитьЗначениеИзОтветаJSON(ТекстJSON) Экспорт

    ЧтениеJSON      = Новый ЧтениеJSON;

    ЧтениеJSON.УстановитьСтроку(ТекстJSON);

    Значение        = ПрочитатьJSON(ЧтениеJSON, Истина);

    Возврат Значение;

КонецФункции

Функция ПолучитьСтрокуJSON(Значение) Экспорт

    ЗаписьJSON = Новый ЗаписьJSON;
    ЗаписьJSON.УстановитьСтроку();
    ЗаписатьJSON(ЗаписьJSON, Значение);

    Возврат ЗаписьJSON.Закрыть();

КонецФункции
```

6.2 web client (html/js)

Пример реалзации websocket клиента в ПолеHTMLДокумента или на web-странице корпоративного портала

`html_client()`

Описание: web-страница для подключения к 1С

```
<!DOCTYPE html>
<meta charset="utf-8" />
<title>WebSocket Test</title>
<script language="javascript" type="text/javascript">

    var wsUri = "ws://127.0.0.1:9098";
    var output;

    function init() {
        output = document.getElementById("output");
        testWebSocket();
    }

    function testWebSocket() {
        websocket = new WebSocket(wsUri);
        websocket.onopen = function (evt) { onOpen(evt) };
        websocket.onclose = function (evt) { onClose(evt) };
        websocket.onmessage = function (evt) { onMessage(evt) };
        websocket.onerror = function (evt) { onError(evt) };
    }

    function onOpen(evt) {
        writeToScreen("CONNECTED");
        doSend("WebSocket rocks");
    }

    function onClose(evt) {
        writeToScreen("DISCONNECTED");
    }

    function onMessage(evt) {
        writeToScreen('<span style="color: blue;">RESPONSE: ' + evt.data + '</span>');
        websocket.close();
    }

    function onError(evt) {
        writeToScreen('<span style="color: red;">ERROR:</span> ' + evt.data);
    }

    function doSend(message) {
        writeToScreen("SENT: " + message);
        websocket.send(message);
    }

    function writeToScreen(message) {
        var pre = document.createElement("p");
        pre.style.wordWrap = "break-word";
        pre.innerHTML = message;
        output.appendChild(pre);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    }

    window.addEventListener("load", init, false);

</script>

<h2>WebSocket Test</h2>

<div id="output"></div>
```

6.3 Другие кейсы

- Сервер обработки сообщений

  Регламентное задание с расписанием повторять каждую 1 секунду запускается на сервере, прослушивает порт и принимает соединения.

  Если обработка входящих сообщений занимает продолжительное время - запускаем фоновое задание обработки и с ИД клиента. Затем возвращаем результат клиенту, если это требуется.

Глава 7

Другое

- genindex
- modindex
- search

Алфавитный указатель

Символы

H

M

S