О применении библиотеки FastScript в своих проектах. Часть-2 «Входные и выходные параметры FS-скрипта (при информационном обмене с внешними инициаторами)»

Оглавление

1 Введение

Следует отметить, что:

1. Соглашения, сокращения, термины (и их определения) приведены в документе [2].
2. Автор излагает свой собственный подход к применению библиотеки FastScript, совершенно не претендуя на «истину в последней инстанции».

В ряде случаев, когда FastScript применяется в гибких ИС, тексты FS-скриптов хранятся в таблицах БД и загружаются для выполнения периодически, согласно принятого регламента, или ситуативно, по «требованию» внешних инициаторов информационного обмена.

«Требования» инициаторов информационного обмена могут включать некие исходные данные, которые должны быть учтены при выполнении FS-скрипта (входные параметры FSI), и возвращаемые данные, полученные в результате выполнения FS-скрипта (вЫходные параметры FSI).

Входные параметры FSI могут быть преобразованы в константы FastScript, а вЫходные параметры FSI – в переменные FastScript.

Т.е., некий функционал (из состава Обвязки) конвертирует поступившие входные параметры – в константы FastScript, а вЫходные параметры – в переменные FastScript.

А затем, соответствующий FS-скрипт выгружается из БД и (используя компонент `TfsScript` библиотеки FastScript) запускается на выполнение.

После того, как FS-скрипт выполнен, значения соответствующих переменных конвертируются Обвязкой в вЫходные параметры (для предоставления их инициатору информационного обмена в соответствии с требованиями).

См. рисунки ниже.

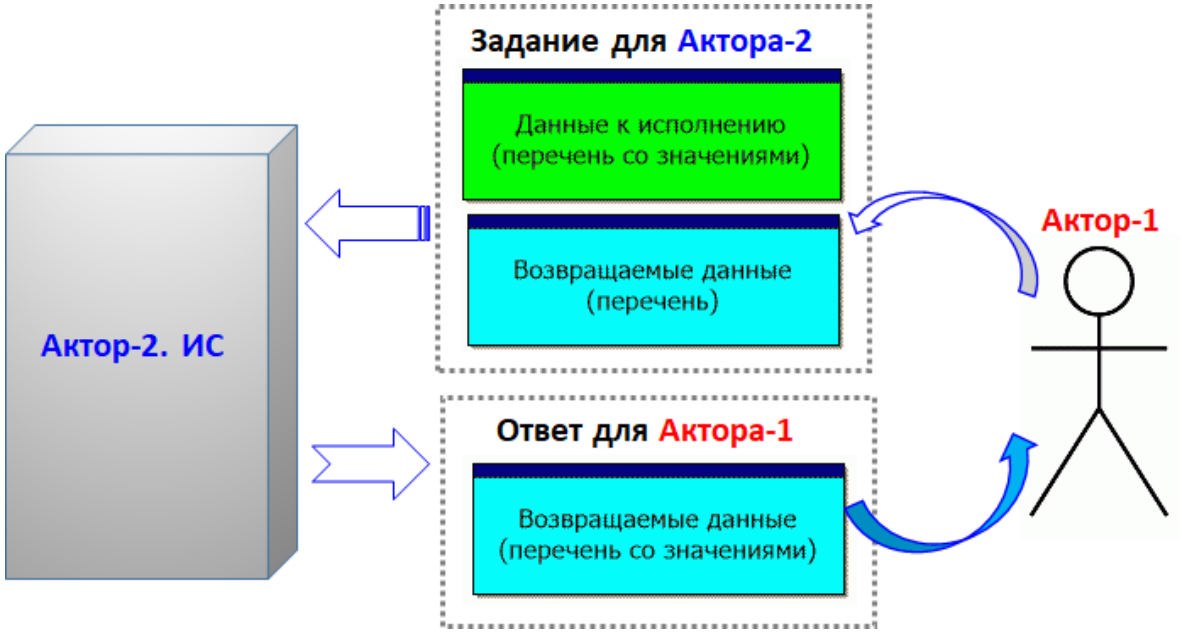


Рисунок 1 – Схема, иллюстрирующая информационный обмен

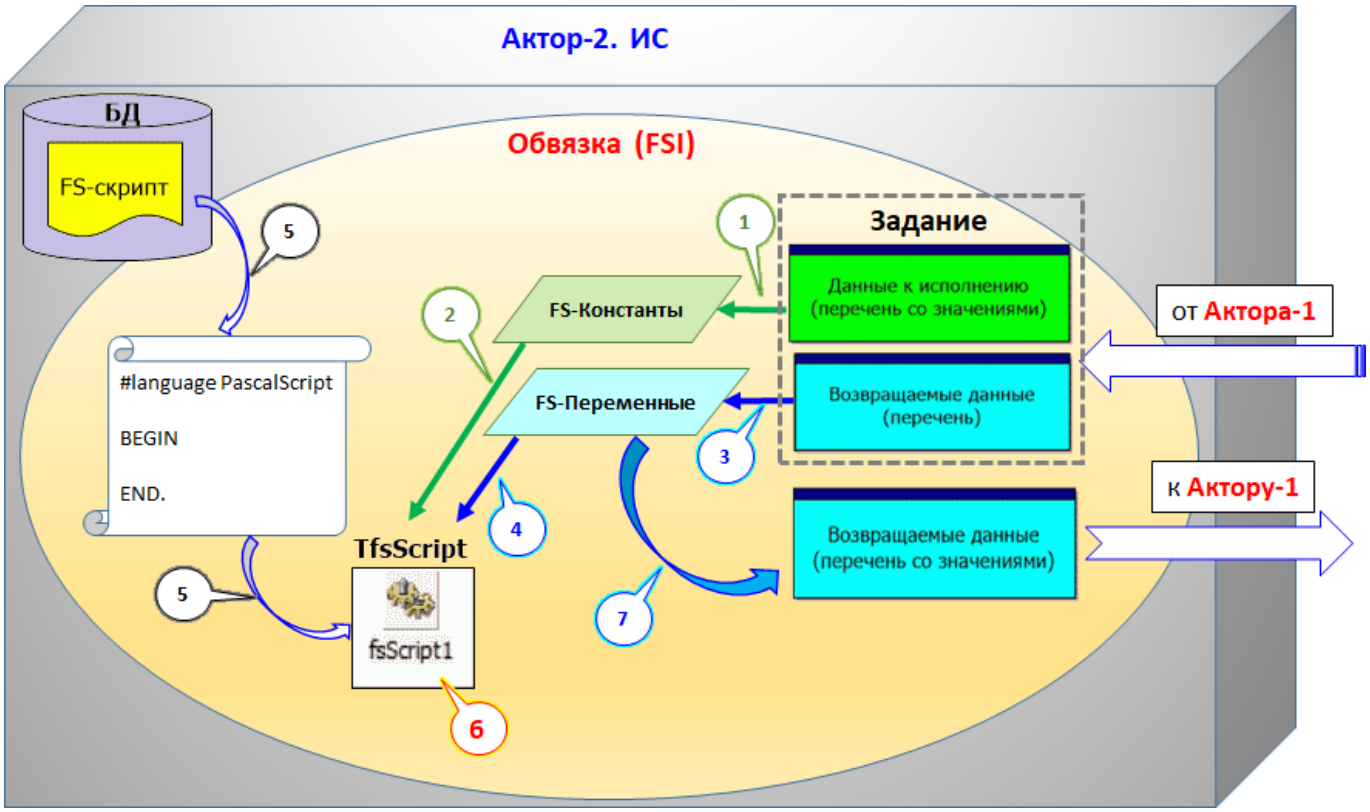


Рисунок 2 – Схема, иллюстрирующая обработку данных

Обозначение выносок на рисунке 2:

Выноской-1 обозначен процесс анализа и нормализации задания (входные данные для исполнения, как входные параметры для FSI) от инициатора информационного обмена (Актор-1).

Выноской-2 обозначен процесс конвертации входных параметров FSI в перечень констант для **`fsScript`** (см. расширение функционала FastScript в документе [2]) и «загрузка» их в **`fsScript`**.

Выноской-3 обозначен процесс анализа и нормализации перечня возвращаемых инициатору информационного обмена данных (как вЫходные параметры FSI).

Выноской-4 обозначен процесс конвертации перечня вЫходных параметров FSI в перечень переменных для **`fsScript`** (см. расширение функционала FastScript в документе [2]) и «загрузка» их в **`fsScript`**.

Выноской-5 обозначен процесс выгрузки соответствующего FS-скрипта из БД и загрузка его в **`fsScript`**.

Выноской-6 обозначен процесс выполнения FS-скрипта (здесь может быть реализован, например, как алгоритм обработки информации, так и алгоритм управления программно-аппаратными комплексами).

Выноской-7 обозначен процесс конвертации переменных FS-скрипта в перечень возвращаемых (к инициатору информационного обмена) данных.

Существует далеко не один вариант реализации информационного взаимодействия между Акторами (см. рисунки 1 и 2).

Это могут быть, например, взаимодействия:

- регламентируемые временными рамками (периодическое выполнение каких-либо расчетов и «выкладывание» результатов расчетов в «общий доступ» на уровне БД);
- по заданию от инициатора информационного взаимодействия с использованием соответствующих таблиц БД;
- по заданию от инициатора информационного взаимодействия в реальном времени.

Способ (в контексте использования FastScript) принципиального значения не имеет.

«Лучше один раз увидеть, чем десять раз услышать».

Поэтому, в рамках данной статьи приведен простой иллюстрирующий пример в варианте «по заданию от инициатора информационного взаимодействия в реальном времени».

Важно!

На рисунке 1 в информационный поток от Актора-1 к Актору-2 (задание) включен и перечень «возвращаемых данных».

В принципе, можно ограничиться только «данными к исполнению».

Но в общем случае может сложиться ситуация, когда в результате каких-то расчетов (на уровне FS-скрипта Актора-2) будет получено существенно «много» данных.

И далеко не все они могут быть необходимы Актору-1.

Включение в информационный поток перечень «возвращаемых данных» решает эту проблему.

2 Практическое применение библиотеки FastScript

2.1 Формат данных информационного обмена

Понятно дело, что формат данных при информационном обмене может быть различным. Но в данном случае будет использоваться простейший вариант – «список значений» (`TStrings`).

2.1.1 Входные параметры для FSI (формат данных от Актора-1 к Актору-2)

2.1.1.1 «Данные к исполнению»

«Данные к исполнению» (задание от Актора-1) являются входными параметрами для FSI. Перед выполнением FS-скрипта они конвертируются Оберткой в список соответствующих FS-констант.

Формат списка значений:

ИмяПараметра=**ЗначениеПараметра**

Важно!

1. `ИмяПараметра` – должно быть корректным с точки зрения допустимости идентификаторов констант в FastScript;
2. `ЗначениеПараметра` – всегда строка (`string`) без специальных символов (включая: `#9, #10, #13, «равно»`).

Примеры:

```
Планета=Марс
Вес_на_Земле=100
Маска_файлов=D:\ttt\*.*
```

2.1.1.2 «Возвращаемые данные (перечень)»

«Перечень возвращаемых данных» (задание от Актора-1) также являются входными параметрами для FSI. Перед выполнением FS-скрипта они конвертируются Оберткой в список соответствующих FS-переменных.

После выполнения FS-скрипта, этот список заполняется значениями соответствующих FS-переменных и возвращается от Актора-2 к Актору-1.

Формат списка значений:

ИмяПараметра= **ЗначениеПараметра**

Важно!

1. `ИмяПараметра` – должно быть корректным с точки зрения допустимости идентификаторов переменных в FastScript;
2. `ЗначениеПараметра` – всегда строка (`string`) без специальных символов (включая: `#9, #10, #13, «равно»`). `ЗначениеПараметра` НЕ используется, поэтому есть смысл присваивать значению параметра просто **пробел** или какой-то иной **символ** (без фанатизма).

Примеры:

```
Результат_выполнения=<пробел>
Масса=?
Вес_на_планете=?
```

2.1.2 Выходные параметры от FSI (формат возвращаемых к Актору-1 данных)

«Возвращаемые данные со значениями» (информационный поток от Актора-2 к Актору-1) являются вЫходными параметрами от FSI.

2.1.2.1 «Возвращаемые данные со значениями»

После выполнения FS-скрипта, список «Возвращаемые данные со значениями» заполняется значениями соответствующих FS-переменных и возвращается от Актора-2 к Актору-1.

Формат списка значений – см. раздел 2.1.1.2.

Важно!

В некоторых случаях Актору-1 возвращается просто текст (например, описАние FS-скрипта).

Примеры:

```
Результат_выполнения=Ok
Масса=10.1971621297793
Вес_на_планете=38.5
```

Пример – описание FS-скрипта:

```
Скрипт позволяет пересчитать вес (тела на Земле)
для условий других планет Солнечной Системы.
Включая Солнце и Луну.
```

2.2 Размещение FS-скриптов

Тексты FS-скриптов хранятся в таблице БД **`SQLite`**.

Структуру таблицы – см. в разделе 5.1 (Приложение-1).

2.3 Используемая технология: DataSnap

При разработке иллюстрирующего примера использована технология DataSnap (см. рисунок 3).

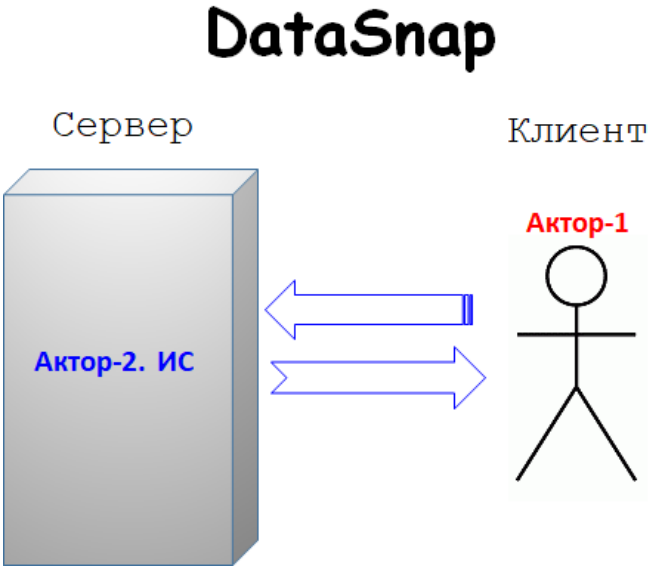


Рисунок 3 – DataSnap Клиент–Сервер

Весь специальный функционал, связанный с использованием библиотеки FastScript, размещен на DataSnap-Сервере (Актор-2).

DataSnap-Клиент (Актор-1) вызывает соответствующие методы DataSnap-Сервера.

На DataSnap-Клиенте реализован функционал, связанный с формированием «Задания» (исходных данных, необходимых для выполнения FS-скриптов) и визуализацией возвращаемых от Актора-2 данных.

Замечания:

1. О технологии DataSnap есть достаточно много информации в Интернете, поэтому здесь дублировать нет смысла.
2. Процесс создания Сервера и Клиента детально рассмотрен в статье [3], размещенной на сайте roamer55.ru.
3. К данной статье прилагаются исходные тексты соответствующих проектов (иллюстрирующий пример).

Важно!

Поскольку это иллюстрирующий пример, то DataSnap-Сервер и DataSnap-Клиент взаимодействуют друг с другом в рамках **`localhost`** (т.е., должны быть запущены на одном компьютере).

2.4 DataSnap-Сервер (Actor-2)

Общие замечания:

1. Тексты FS-скриптов хранятся в таблице БД **`SQLite`** – см. раздел 5.1 (Приложение-1).
2. В разделе 5.2 (Приложение-2) приведены исходники основных (знАчимых) функций, остальные следует смотреть в исходниках прилагаемого иллюстрирующего примера (или в соответствующих HTML-файлах).
3. **Порт** (используемый для информационного обмена): **`50016`**.

На рисунке ниже приведен внешний вид главной формы DataSnap-Сервера (в режиме разработки).

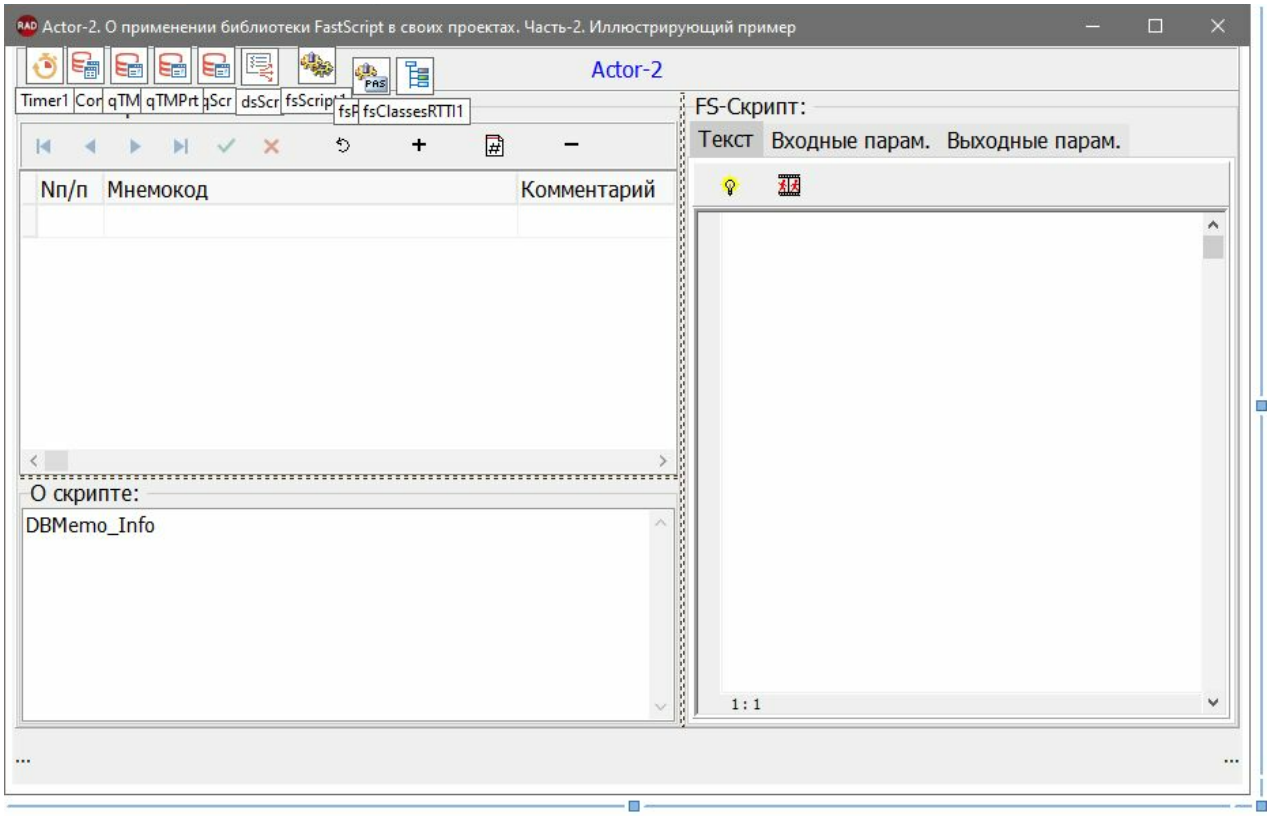


Рисунок 4 – Внешний вид главной формы DataSnap-Сервера (в режиме разработки)

Методы DataSnap-сервера (см., также, рисунок 5):

```
//Список FS-скриптов
function FS_Scripts_List:string;
//FS-скрипт: комментарий к скрипту
function FS_Script_Comment(sMnemo:string):string;
//FS-скрипт: описание к скрипта
function FS_Script_Info(sMnemo:string):string;
//FS-скрипт: список входных параметров FSI
function FS_Script_ParamsIn(sMnemo:string):string;
//FS-скрипт: список вЫходных параметров FSI
function FS_Script_ParamsOut(sMnemo:string):string;
//FS-скрипт: Выполнить
function FS_Script_Execute(sMnemo:string; sParamsIn:string;
sParamsOut:string):string;
```

```
//-----------------------------------
//"Наши" методы...    :-)
//Список FS-скриптов
function FS_Scripts_List:string;
//FS-скрипт: комментарий к скрипту
function FS_Script_Comment(sMnemo:string):string;
//FS-скрипт: описание к скрипта
function FS_Script_Info(sMnemo:string):string;
//FS-скрипт: список входных параметров FSI
function FS_Script_ParamsIn(sMnemo:string):string;
//FS-скрипт: список вЫходных параметров FSI
function FS_Script_ParamsOut(sMnemo:string):string;
//FS-скрипт: Выполнить
function FS_Script_Execute(sMnemo:string; sParamsIn:string; sParamsOut:string):string;
//-----------------------------------
```

Рисунок 5 – Методы DataSnap-Сервера

Реализация этих методов – это, по сути, вызов соответствующих функций из раздела 5.2 (Приложение-2).

Все технические детали – см. исходные тексты иллюстрирующего примера.

2.5 DataSnap-Клиент (Actor-1)

Как было сказано выше: DataSnap-Клиент (Актор-1) вызывает соответствующие методы DataSnap-Сервера. На DataSnap-Клиенте реализован функционал, связанный с формированием «Задания» (исходных данных, необходимых для выполнения FS-скриптов) и визуализацией возвращаемых от Актора-2 данных.

На рисунке 6 приведены основные параметры DataSnap-Клиента.

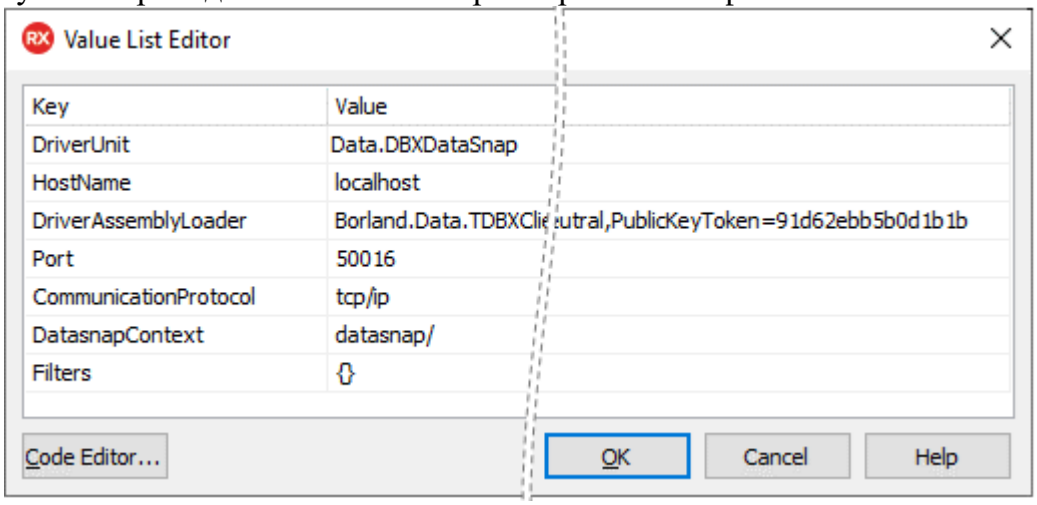


Рисунок 6 – Основные параметры DataSnap-Клиента

На рисунке ниже приведен внешний вид главной формы DataSnap-Клиента (в режиме разработки).

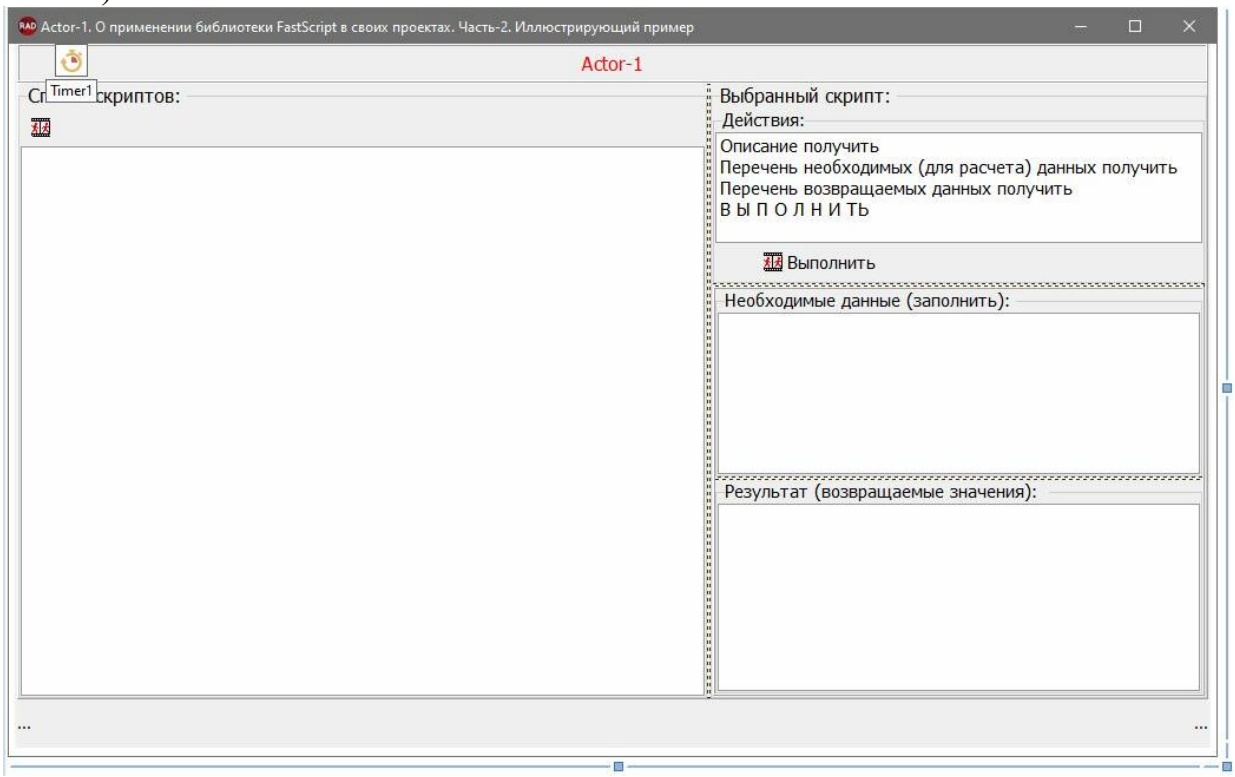


Рисунок 7 – Внешний вид главной формы DataSnap-Клиента (в режиме разработки)

Все технические детали – см. исходные тексты иллюстрирующего примера.

3 Иллюстрирующий пример. Основные операции

К статье прилагаются, также, готовые EXE-модули DataSnap-Сервера и DataSnap-Клиента:

`FS_02_io_param_SRV.exe` – DataSnap-Сервер;
`FS_02_io_param_CLN.exe` – DataSnap-Клиент;
`FS_02_io_param.db` – SQLite БД (где хранятся FS-скрипты).

Общие замечания:

1. Сервер и Клиент должны быть запущены на одном компьютере.
2. Сервер должен быть запущен ПЕРЕД Клиентом.
3. Порт (**`50016`**) должен быть доступен для использования.
4. Никакой «инсталляции» не требуется:
   «вытащил» из ZIP-архива и «танцуй с бубном».
5. Лучше создать отдельную папку и все три файла (см. выше) скопировать в нее.
6. В таблице `fs_scripts` БД SQLite (см. файл `FS_02_io_param.db`) уже хранятся 3 простых скрипта.
7. На рисунках 8 – 10 приведены основные элементы управления GUI DataSnap-Сервера и DataSnap-Клиента.

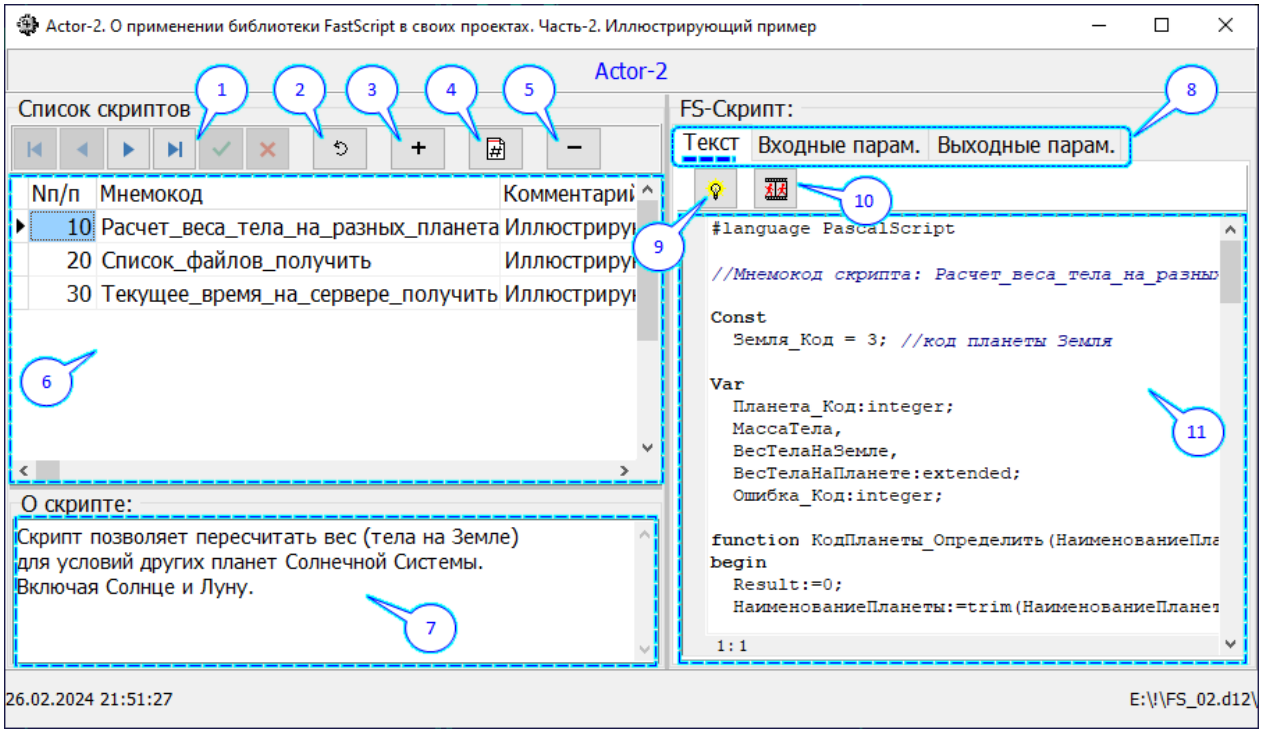


Рисунок 8 – Внешний вид главной формы DataSnap-Сервера

В таблице 1 приведено обозначение выносок на рисунке 8:

Таблица 1 – Обозначение выносок на рисунке 8

Выноска	**Обозначение**
1	**2**
1	Навигатор по таблице БД (список FS-скриптов)
2	Кнопка «Обновить список FS-скриптов на Экране»
3	Кнопка «Добавить новый FS-скрипт»
4	Кнопка «Перенумеровать список FS-скриптов с шагом 10»
5	Кнопка «Удалить выбранный FS-скрипт»

Выноска	Обозначение
1	**2**
6	Таблица (список) FS-скриптов
7	ОписАние выбранного FS-скрипта
8	Область кнопок, для переключения на соответствующую вкладку
9	Кнопка «Компилировать выбранный FS-скрипт» (в режиме разработки). Находится на вкладке «Текст»
10	Кнопка «Выполнить выбранный FS-скрипт» (в режиме разработки). Находится на вкладке «Текст»
11	Редактор FS-скриптов (`TfsSyntaxMemo`). Находится на вкладке «Текст»

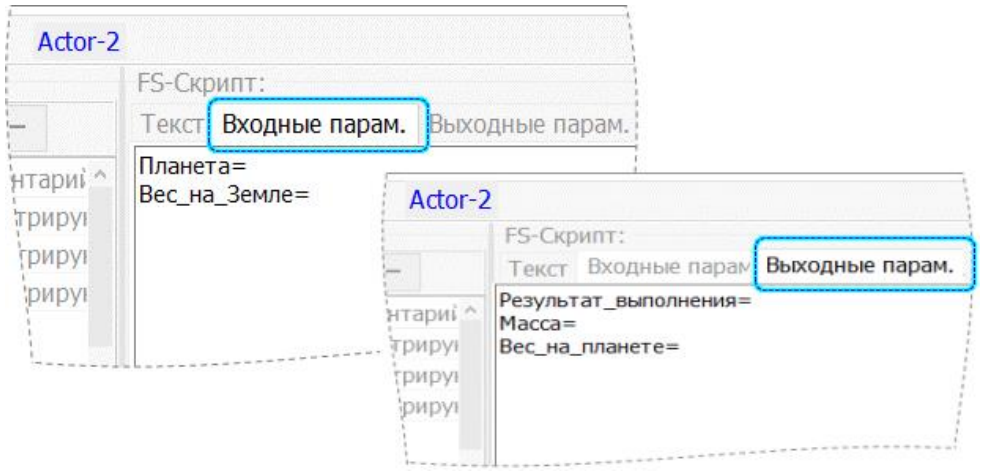


Рисунок 9 – Вкладки «Входные парам.» и «Выходные парам.» главной формы DataSnap-Сервера

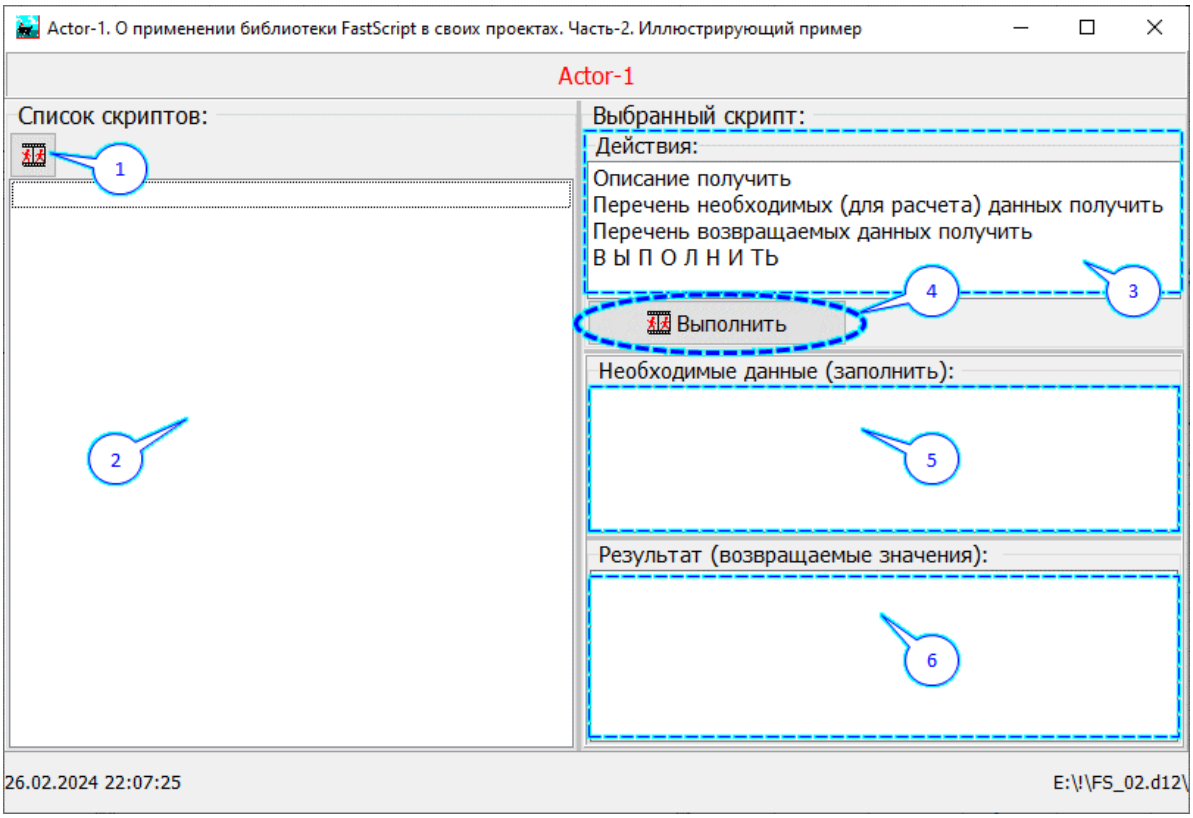


Рисунок 10 − Внешний вид главной формы DataSnap-Клиента

В таблице 2 приведено обозначение выносок на рисунке 10:

Таблица 2 − Обозначение выносок на рисунке 10

Выноска	**Обозначение**
1	**2**
1	Кнопка «Получить список FS-скриптов»
2	Список FS-скриптов
3	Список доступных команд
4	Кнопка «Выполнить выбранную команду»
5	Перечень данных к исполнению (см. рисунок 1 и раздел 2.1.1.1)
6	Возвращаемые данные (см. рисунок 1 и разделы 2.1.1.2, 2.1.2.1)

3.1 Начало «работы»

1. Запустить DataSnap-Сервер (`FS_02_io_param_SRV.exe`).
2. Запустить DataSnap-Клиент (`FS_02_io_param_CLN.exe`).
3. Разместить окна Приложений на Экране так, чтобы было удобно.

3.2 Операции на Клиенте (`FS_02_io_param_CLN.exe`)

3.2.1 Получить список FS-скриптов

1. Нажать на кнопку, обозначенную выноской-1 на рисунке 10.

Результат: список FS-скриптов (выноска-2 на рисунке 10) будет актуализирован.

3.2.2 Выбрать FS-скрипт в списке доступных

1. Выбрать мышкой соответствующую строку в списке FS-скриптов (выноска-2 на рисунке 10).

3.2.3 Получить информацию о выбранном FS-скрипте

1. В списке доступных действий (выноска-3 на рисунке 10) выбрать строку «Описание получить».
2. Нажать на кнопку, обозначенную выноской-4 на рисунке 10.

Результат: в области, отмеченной выноской-6 на рисунке 10, появится описание выбранного FS-скрипта.

3.2.4 Получить «перечень данных к исполнению» для выбранного FS-скрипта

1. В списке доступных действий (выноска-3 на рисунке 10) выбрать строку «Перечень необходимых (для расчета) данных получить».
2. Нажать на кнопку, обозначенную выноской-4 на рисунке 10.

Результат: в области, отмеченной выноской-5 на рисунке 10, появится перечень соответствующих данных.

3.2.5 Получить «перечень возвращаемых данных» для выбранного FS-скрипта

1. В списке доступных действий (выноска-3 на рисунке 10) выбрать строку «Перечень возвращаемых данных получить».
2. Нажать на кнопку, обозначенную выноской-4 на рисунке 10.

Результат: в области, отмеченной выноской-6 на рисунке 10, появится перечень соответствующих данных.

3.2.6 Выполнить выбранный FS-скрипт

1. Выбрать соответствующий FS-скрипт (см. раздел 3.2.2).
2. Выполнить действия, указанные в разделе 3.2.4.
3. Заполнить корректными значениями соответствующие параметры (см. выноску-5 на рисунке 10).
4. Выполнить действия, указанные в разделе 3.2.5.
5. В списке доступных действий (выноска-3 на рисунке 10) выбрать строку «В Ы П О Л Н И Т Ь».
6. Нажать на кнопку, обозначенную выноской-4 на рисунке 10.

Результат: в области, отмеченной выноской-6 на рисунке 10, появится перечень соответствующих данных.

3.3 Операции на Сервере (`FS_02_io_param_SRV.exe`)

3.3.1 Обновить список FS-скриптов на Экране

1. Нажать на кнопку, обозначенную выноской-2 на рисунке 8.

Результат: таблица (список) FS-скриптов (выноска-6 на рисунке 8) «Переоткроется».

3.3.2 Перенумеровать список FS-скриптов с шагом 10

1. Нажать на кнопку, обозначенную выноской-4 на рисунке 8.

Результат: строки в таблице (списке) FS-скриптов (выноска-6 на рисунке 8) перенумеруются с шагом 10.

3.3.3 Создать новый FS-скрипт

1. Нажать на кнопку, обозначенную выноской-3 на рисунке 8.
2. В диалоговом окне ввести мнемокод скрипта (уникальное имя).
3. Нажать на кнопку OK.
4. Заполнить описание скрипта (выноска-7 на рисунке 8).
5. Ввести исходный текст скрипта (выноска-11 на рисунке 8).
6. Корректно (в соответствии с исходным текстом скрипта) создать списки входных и выходных параметров (см. рисунок 9).

3.3.4 Выбрать FS-скрипт в таблице (списке)

1. Выбрать мышкой соответствующую строку в таблице (списке) FS-скриптов (выноска-6 на рисунке 8).

3.3.5 Удалить выбранный FS-скрипт

1. Нажать на кнопку, обозначенную выноской-5 на рисунке 8.
2. Нажать на кнопку OK в ответ на запрос о подтверждении.

3.3.6 Редактировать выбранный FS-скрипт

Выполнить соответствующие изменения (отредактировать содержание):
- в строке таблицы (списка) скриптов (выноска-6 на рисунке 8);
- в описании скрипта (выноска-7 на рисунке 8);
- в исходном тексте скрипта (выноска-11 на рисунке 8);
- в списках входных и вЫходных параметров (см. рисунок 9).

3.3.7 Отладка выбранного FS-скрипта

1. Корректно (в соответствии с исходным текстом скрипта) ввести значения входных параметров (см. рисунок 9).
2. На вкладке «Текст» нажать на кнопку «Компиляция» (выноска-9 на рисунке 8).
3. Исправить ошибки в исходном тексте скрипта (если они есть).
4. Нажать на кнопку «Выполнение» (выноска-10 на рисунке 8).
5. Перейти на вкладку «Выходные парам.» (см. рисунок 9) и произвести контроль результата выполнения скрипта.

4 Использованные источники

1. «FastScript. Библиотека скриптов. Руководство разработчика» (опубликовано на сайте Разработчика)
2. О применении библиотеки FastScript в своих проектах. Часть-1 «Расширение функционала» (опубликована на сайте roamer55.ru, имя файла: `01_FS_exp_func.pdf`)
3. Использование DataSnap-технологии на примере разработки комплекса взаимодействующих приложений (ОС Windows и ОС Android) в среде Delphi 10.2 Tokyo. Последовательность действий (опубликована на сайте roamer55.ru, имя файла: `spBallCollision.pdf`)

5 Приложения

5.1 Приложение-1. Структура таблицы БД для размещения FS-скриптов

Тексты FS-скриптов хранятся в таблице БД **SQLite** следующей структуры (см. рисунок ниже):

Структура SQLite БД: FS_02_io_param.db

```
CREATE TABLE fs_scripts -- список FS-скриптов
(
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, -- уникальный ID
npp INTEGER, -- порядковый номер скрипта в списке
mnemocode VARCHAR(100), -- мнемокод FS-скрипта
mnemocode_lower VARCHAR(100), -- мнемокод FS-скрипта (все символы - в нижнем регистре)
comment VARCHAR(100), -- комментарий
fs_script TEXT, -- текст FS-скрипта
params_in TEXT, -- входные параметры (константы FS-скрипта)
params_out TEXT, -- выходные параметры (переменные FS-скрипта)
info TEXT -- краткое описание
);

CREATE UNIQUE INDEX fs_scripts_mnemocode_lower ON fs_scripts(mnemocode_lower ASC);
CREATE INDEX fs_scripts_npp ON fs_scripts(npp ASC);
```

Рисунок 11 – Таблица **fs_scripts** БД SQLite

Имя БД: **FS_02_io_param.db**.

Файл БД должен находиться в той же папке, что и программа **FS_02_io_param_SRV.exe** (DataSnap-Сервер).

5.2 Приложение-2. Специальный функционал на стороне DataSnap-сервера

Ниже приведены исходники только основных (знАчимых) фунций, остальные следует смотреть в исходниках прилагаемого иллюстрирующего примера (или в соответствующих HTML-файлах).

5.2.1 Конвертация входных и вЫходных параметров FSI

Функции, исходники которых приведены в этом разделе, выполняет основную роль при конвертации соответствующих данных информационного обмена между Акторами в FS-константы и FS-переменные.

5.2.1.1 Функция `MyParams_AddToFS`

Функция **`MyParams_AddToFS`** предназначена для конвертации входных и/или вЫходных параметров FSI в FS-константы и/или FS-переменные.

Исходник этой функции приведен на рисунке ниже.

```
function MyParams_AddToFS(fsScr: TfsScript; ItsVar:boolean; ListParams:TStrings):boolean;
//Добавить входные или вЫходные параметры FSI, как константы/переменные в FastScript
//ListParams - список входных или вЫходных параметров
//ItsVar - флаг:
//         если =FALSE, то ListParams список входных параметров,
//         которые добавляются, как FS-константы;
//         если =TRUE, то ListParams список вЫходных параметров,
//         которые добавляются, как FS-переменные.
Var
  i:integer;
  sName : string;
  sType:string;
  sVal:string;
begin
  Result:=false;
  if Assigned(fsScr) then begin
     if Assigned(ListParams) then begin
        Result:=true;
        //---------------------------------
        //Удалить пустые строки из списка параметров
        TStrings_DeleteEmptyLine(ListParams);
        //---------------------------------
        //---------------------------------
        //Конвертация списка параметров
        //в FS-константы или FS-переменные
        i:=-1;
        while i<(ListParams.Count-1) do
         begin
           i:=i+1;
           //sName - идентификатор (имя) константы или переменной
           sName:=trim(ListParams.Names[i]);
           if sName<>'' then begin
              sType:='string'; //тип - всегда String
              //sVal - значение константы или переменной
              sVal:=trim(ListParams.Values[sName]);
              //.............................................
              //Добавить переменную или константу
              if not fs_VarConst_Add(fsScr, sName, ItsVar, sType, sVal) then begin
                 Result:=false;
                 i:=(ListParams.Count+1);
              end;
              //.............................................
           end;
        end;
        //---------------------------------
     end;
  end;
end;
```

Рисунок 12 – Функция `MyParams_AddToFS`

Как видно (см. рисунок выше), функция `MyParams_AddToFS` вызывает для конвертации каждого параметра функцию `fs_VarConst_Add`.

5.2.1.2 Функция `fs_VarConst_Add`

Функция **`fs_VarConst_Add`** предназначена для добавления одного, конкретного входного или вЫходного параметра FSI – в FS-константу или FS-переменную.

Исходник этой функции приведен на рисунке ниже.

```
function fs_VarConst_Add(fsScr: TfsScript;
                         sName : string;
                         ItsVar:boolean;
                         sType:string;
                         sVal:string=''): boolean;
//Добавить константу или переменную в FastScript
//  ItsVar - см. функцию MyParams_AddToFS
//  sName - идентификатор (имя) FS-константы или FS-переменной
//  sType - тип константы или переменной
//  sVal - значение константы илли переменной
begin
  Result:=false;
  if Assigned(fsScr) then begin
     sName:=trim(sName);
     sType:=trim(sType);
     if (sName<>'') and (sType<>'') then begin
        if sName[1]<>'*' then begin
           //Проверка - существует ли константа или переменная
           //с таким же идентификатором
           if fs_GetIndex(fsScr, sName)<0 then begin
              if ItsVar then begin
                 //добавляем FS-переменную
                 sVal:=''; //для переменных всегда "пустое" значение
                 if AnsiUpperCase(sType)=AnsiUpperCase('String') then begin
                    fsScr.AddVariable(sName, sType,  sVal);
                    Result:=true;
                 end;
                 if AnsiUpperCase(sType)=AnsiUpperCase('Boolean') then begin
                    fsScr.AddVariable(sName, sType,  spStrToBool(trim(sVal),false));
                    Result:=true;
                 end;
                 if AnsiUpperCase(sType)=AnsiUpperCase('Integer') then begin
                    fsScr.AddVariable(sName, sType,  StrToIntDef(trim(sVal),0));
                    Result:=true;
                 end;
                 if AnsiUpperCase(sType)=AnsiUpperCase('Extended') then begin
                    fsScr.AddVariable(sName, sType,  StrToFloatDef(trim(sVal),0.0));
                    Result:=true;
                 end;
              end
              else begin
                 //добавляем FS-константу
                 if AnsiUpperCase(sType)=AnsiUpperCase('String') then begin
                    fsScr.AddConst(sName, sType,  sVal);
                    Result:=true;
                 end;
                 if AnsiUpperCase(sType)=AnsiUpperCase('Boolean') then begin
                    fsScr.AddConst(sName, sType,  spStrToBool(trim(sVal),false));
                    Result:=true;
                 end;
                 if AnsiUpperCase(sType)=AnsiUpperCase('Integer') then begin
                    fsScr.AddConst(sName, sType,  StrToIntDef(trim(sVal),0));
                    Result:=true;
                 end;
                 if AnsiUpperCase(sType)=AnsiUpperCase('Extended') then begin
                    fsScr.AddConst(sName, sType,  StrToFloatDef(trim(sVal),0.0));
                    Result:=true;
                 end;
              end;
           end;
        end
        else begin
           Result:=true;
        end;
     end;
  end;
end;
```

Рисунок 13 – Функция `fs_VarConst_Add`

5.2.1.3 Функция MyParams_VarFromFS

Функция MyParams_VarFromFS предназначена для конвертации FS-переменных в вЫходные параметры FSI (после выполнения скрипта) для возврата их инициатору информационного обмена.

Исходник функции MyParams_VarFromFS приведен ниже (см., также, рисунок 14):

```
function MyParams_VarFromFS(fsScr: TfsScript; ListParams:TStrings):boolean;
//Прочитать значения переменных FastScript в вЫходные параметры FSI
Var
  i:integer;
  Sx, sName : string;
  sType:string;
  sVal:string;
begin
  Result:=false;
  if Assigned(fsScr) then begin
     if Assigned(ListParams) then begin
        Result:=true;
        //-----------------------------------------------
        //конвертация переменных в вЫходные параметры FSI
        i:=-1;
        while i<(ListParams.Count-1) do
         begin
           i:=i+1;
           sName:=trim(ListParams.Names[i]);
           if sName<>'' then begin
              ListParams.Values[sName]:=trim(fsScr.Variables[sName]);
           end
           else begin
              ListParams[i]:='';
           end;
        end;
        //-----------------------------------------------
        //-----------------------------------------------
        //Удаление пустых строк из ListParams
        TStrings_DeleteEmptyLine(ListParams);
        //-----------------------------------------------
     end;
  end;
end;
```

```
function MyParams_VarFromFS(fsScr: TfsScript; ListParams:TStrings):boolean;
//Прочитать значения переменных FastScript в вЫходные параметры FSI
Var
  i:integer;
  Sx, sName : string;
  sType:string;
  sVal:string;
begin
  Result:=false;
  if Assigned(fsScr) then begin
     if Assigned(ListParams) then begin
        Result:=true;
        //-----------------------------------------------
        //конвертация переменных в вЫходные параметры FSI
        i:=-1;
        while i<(ListParams.Count-1) do
         begin
           i:=i+1;
           sName:=trim(ListParams.Names[i]);
           if sName<>'' then begin
              ListParams.Values[sName]:=trim(fsScr.Variables[sName]);
           end
           else begin
              ListParams[i]:='';
           end;
        end;
        //-----------------------------------------------
        //-----------------------------------------------
        //Удаление пустых строк из ListParams
        TStrings_DeleteEmptyLine(ListParams);
        //-----------------------------------------------
     end;
  end;
end;
```

Рисунок 14 – Функция `MyParams_VarFromFS`

5.2.2 Расширение функционала FastScript

5.2.2.1 Функция `fsiScript_MyFuncs_Add`

Исходник функции `fsiScript_MyFuncs_Add`, вызываемой для добавления новых процедур/функций в FastScript:

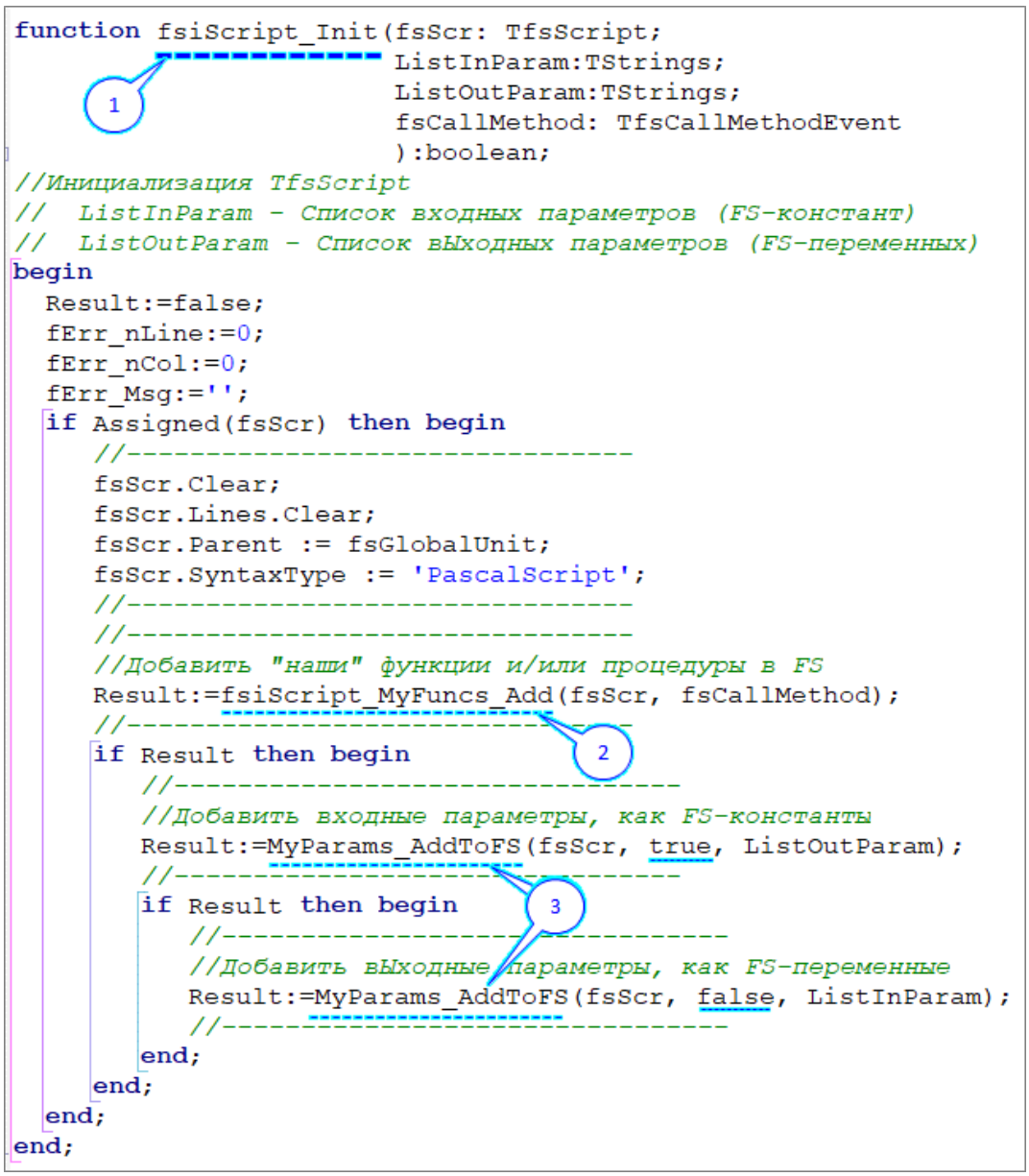
```
function fsiScript_MyFuncs_Add(fsScr: TfsScript;
                               fsCallMethod: TfsCallMethodEvent
                              ):boolean;
//Добавить дополнительные функции в FastScript
begin
  Result:=false;
  if Assigned(fsScr) then begin
     if Assigned(fsCallMethod) then begin
        Result:=true;
        MyFunc_AddToFS(fsScr, 'function','extended',
                       'Строка_в_ВеществЧисло;String_to_Float',
                       'Sx:string;'
                      +'vDef:extended=0.0'
                       ,
                       fsCallMethod);
        MyFunc_AddToFS(fsScr,'function', 'extended',
                       'ОкруглитьДо;RoundTo',
                       'V:extended;iRoundTo:integer=2'
                       ,
                        fsCallMethod);
        MyFunc_AddToFS(fsScr,
                       'function', 'integer',
                       'Файлы_Список_Получить;Files_List_Get',
                       'List: TStrings;'
                      +' FullNameMask: string='+#39+'*.*'+#39+';'
                      +' YesFileNamesOnly: boolean=true'
                       ,
                       fsCallMethod);
     end;
  end;
end;
```

Функция **`MyFunc_AddToFS`** была рассмотрена в документе [2] (там же приведен и ее исходник).

5.2.2.2 Функция `fsiScript_Init`

Исходник функции **`fsiScript_Init`**, вызываемой для инициализации компонента **`TfsScript`** (см., также, рисунок 15):

```
function fsiScript_Init(fsScr: TfsScript;
                        ListInParam:TStrings;
                        ListOutParam:TStrings;
                        fsCallMethod: TfsCallMethodEvent
                        ):boolean;
//Инициализация TfsScript
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
begin
  Result:=false;
  fErr_nLine:=0;
  fErr_nCol:=0;
  fErr_Msg:='';
  if Assigned(fsScr) then begin
     //--------------------------------
     fsScr.Clear;
     fsScr.Lines.Clear;
     fsScr.Parent := fsGlobalUnit;
     fsScr.SyntaxType := 'PascalScript';
     //--------------------------------
     //--------------------------------
     //Добавить "наши" функции и/или процедуры в FS
     Result:=fsiScript_MyFuncs_Add(fsScr, fsCallMethod);
     //--------------------------------
     if Result then begin
        //--------------------------------
        //Добавить входные параметры, как FS-константы
        Result:=MyParams_AddToFS(fsScr, true, ListOutParam);
        //--------------------------------
        if Result then begin
           //--------------------------------
           //Добавить вЫходные параметры, как FS-переменные
           Result:=MyParams_AddToFS(fsScr, false, ListInParam);
           //--------------------------------
        end;
     end;
  end;
end;
```




```
function fsiScript_Init(fsScr: TfsScript;
                        ListInParam:TStrings;
                        ListOutParam:TStrings;
                        fsCallMethod: TfsCallMethodEvent
                        ):boolean;
//Инициализация TfsScript
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
begin
  Result:=false;
  fErr_nLine:=0;
  fErr_nCol:=0;
  fErr_Msg:='';
  if Assigned(fsScr) then begin
     //--------------------------------
     fsScr.Clear;
     fsScr.Lines.Clear;
     fsScr.Parent := fsGlobalUnit;
     fsScr.SyntaxType := 'PascalScript';
     //--------------------------------
     //--------------------------------
     //Добавить "наши" функции и/или процедуры в FS
     Result:=fsiScript_MyFuncs_Add(fsScr, fsCallMethod);
     //--------------------------------
     if Result then begin
        //--------------------------------
        //Добавить входные параметры, как FS-константы
        Result:=MyParams_AddToFS(fsScr, true, ListOutParam);
        //--------------------------------
        if Result then begin
           //--------------------------------
           //Добавить вЫходные параметры, как FS-переменные
           Result:=MyParams_AddToFS(fsScr, false, ListInParam);
           //--------------------------------
        end;
     end;
  end;
end;
```

Рисунок 15 – Функция Обвязки **fsiScript_Init**

Обозначение выносок на рисунке выше:

Выноской-1 обозначена функция **fsiScript_Init.**

Выноской-2 обозначена функция **fsiScript_MyFuncs_Add** (ее исходник см. выше).

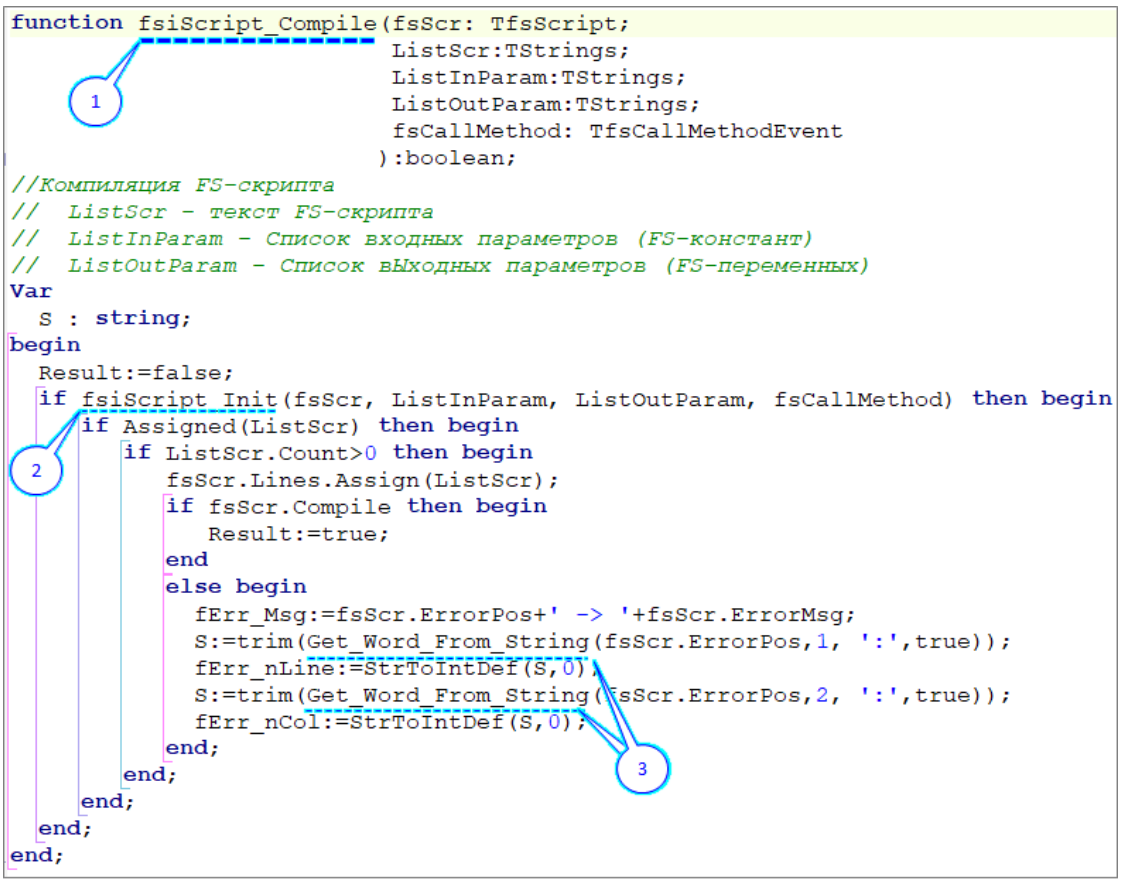
Выноской-3 обозначена вспомогательная функция `Get_Word_From_String` (ее исходник приведен в документе [2]).

5.2.3 Компиляция и выполнение FS-скрипта

5.2.3.1 Функция `fsiScript_Compile`

Исходник функции **`fsiScript_Compile`**, вызываемой при компиляции (контроль ошибок) и выполнении скриптов (см., также, рисунок 16):

```
function fsiScript_Compile(fsScr: TfsScript;
                           ListScr:TStrings;
                           ListInParam:TStrings;
                           ListOutParam:TStrings;
                           fsCallMethod: TfsCallMethodEvent
                          ):boolean;
//Компиляция FS-скрипта
//  ListScr - текст FS-скрипта
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
Var
  S : string;
begin
  Result:=false;
  if fsiScript_Init(fsScr, ListInParam, ListOutParam, fsCallMethod) then begin
     if Assigned(ListScr) then begin
        if ListScr.Count>0 then begin
           fsScr.Lines.Assign(ListScr);
           if fsScr.Compile then begin
              Result:=true;
           end
           else begin
             fErr_Msg:=fsScr.ErrorPos+' -> '+fsScr.ErrorMsg;
             S:=trim(Get_Word_From_String(fsScr.ErrorPos,1, ':',true));
             fErr_nLine:=StrToIntDef(S,0);
             S:=trim(Get_Word_From_String(fsScr.ErrorPos,2, ':',true));
             fErr_nCol:=StrToIntDef(S,0);
           end;
        end;
     end;
  end;
end;
```




```
function fsiScript_Compile(fsScr: TfsScript;
                           ListScr:TStrings;
                           ListInParam:TStrings;
                           ListOutParam:TStrings;
                           fsCallMethod: TfsCallMethodEvent
                          ):boolean;
//Компиляция FS-скрипта
//  ListScr - текст FS-скрипта
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
Var
  S : string;
begin
  Result:=false;
  if fsiScript_Init(fsScr, ListInParam, ListOutParam, fsCallMethod) then begin
    if Assigned(ListScr) then begin
      if ListScr.Count>0 then begin
        fsScr.Lines.Assign(ListScr);
        if fsScr.Compile then begin
          Result:=true;
        end
        else begin
          fErr_Msg:=fsScr.ErrorPos+' -> '+fsScr.ErrorMsg;
          S:=trim(Get_Word_From_String(fsScr.ErrorPos,1, ':',true));
          fErr_nLine:=StrToIntDef(S,0);
          S:=trim(Get_Word_From_String(fsScr.ErrorPos,2, ':',true));
          fErr_nCol:=StrToIntDef(S,0);
        end;
      end;
    end;
  end;
end;
```

Рисунок 16 – Функция Обвязки **`fsiScript_Compile`**

Обозначение выносок на рисунке выше:

Выноской-1 обозначена функция **`fsiScript_Compile.`**

Выноской-2 обозначена функция **`fsiScript_Init`** (ее исходник см. выше).

Выноской-3 обозначена вспомогательная функция `Get_Word_From_String` (ее исходник приведен в документе [2]).

5.2.3.2 Функция `fsiScript_Run`

Исходник функции `fsiScript_Run`, вызываемой при выполнении скриптов в режиме их разработки, приведен ниже (см., также, рисунок 17):

```
function fsiScript_Run(fsScr: TfsScript;
                       ListScr:TStrings;
                       ListInParam:TStrings;
                       ListOutParam:TStrings;
                       fsCallMethod: TfsCallMethodEvent
                      ):boolean;
//Выполнение FS-скрипта (в режиме разработки)
//  ListScr - текст FS-скрипта
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
begin
  //-----------------------------
  //Инициализация fsScr;
  //конвертация входных и вЫходных параметров FSI;
  //компиляция (контроль ошибок) FS-скрипта
  Result := fsiScript_Compile(fsScr,
                              ListScr,
                              ListInParam,
                              ListOutParam,
                              fsCallMethod
                             );
  //-----------------------------
  if Result then begin
     //-----------------------------
     //Выполнение FS-скрипта
     fsScr.Execute;
     //-----------------------------
     //-----------------------------
     //Конвертация FS-переменных в вЫходные параметры FSI
     MyParams_VarFromFS(fsScr, ListOutParam);
     //-----------------------------
  end;
end;
```

```
function fsiScript_Run(fsScr: TfsScript;
                       ListScr:TStrings;
                       ListInParam:TStrings;
                       ListOutParam:TStrings;
                       fsCallMethod: TfsCallMethodEvent
                      ):boolean;
//Выполнение FS-скрипта (в режиме разработки)
//  ListScr - текст FS-скрипта
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
begin
  //-----------------------------
  //Инициализация fsScr;
  //конвертация входных и вЫходных параметров FSI;
  //компиляция (контроль ошибок) FS-скрипта
  Result := fsiScript_Compile(fsScr,
                              ListScr,
                              ListInParam,
                              ListOutParam,
                              fsCallMethod
                             );
  //-----------------------------
  if Result then begin
     //-----------------------------
     //Выполнение FS-скрипта
     fsScr.Execute;
     //-----------------------------
     //-----------------------------
     //Конвертация FS-переменных в выходные параметры FSI
     MyParams_VarFromFS(fsScr, ListOutParam);
     //-----------------------------
  end;
end;
```

Рисунок 17 – Функция `fsiScript_Run`

Исходник функции **`MyParams_VarFromFS`** приведен выше.

5.2.3.3 Функция `fsiScript_Run_RT`

Исходник функции `fsiScript_Run_RT`, вызываемой при выполнении скриптов в реальном времени (в процессе информационного обмена между Акторами), приведен ниже (см., также, рисунок 18):

```
function fsiScript_Run_RT(fsScr: TfsScript;
                          ListScr:TStrings;
                          ListInParam:TStrings;
                          ListOutParam:TStrings;
                          fsCallMethod: TfsCallMethodEvent
                         ):boolean;
//Выполнение FS-скрипта в режиме реального времени
//(при обращении внешнего инициатора информационного обмена)
//  ListScr - текст FS-скрипта
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
Var
  ItsOk:boolean;
begin
  Result := false;
  ItsOk:=false;
  TRY
    //----------------------------
    //Инициализация fsScr;
    //конвертация входных и вЫходных параметров FSI;
    //компиляция (контроль ошибок) FS-скрипта
    Result := fsiScript_Compile(fsScr,
                                ListScr,
                                ListInParam,
                                ListOutParam,
                                fsCallMethod
                               );

    //----------------------------
    if Result then begin
       Result := false;
       //----------------------------
       //Выполнение FS-скрипта
       fsScr.Execute;
       //----------------------------
       //----------------------------
       //Конвертация FS-переменных в выходные параметры FSI
       MyParams_VarFromFS(fsScr, ListOutParam);
       //----------------------------
       ItsOk:=true;
       Result := true;
    end;
  EXCEPT
    if not ItsOk then begin
       //FS вывалился в Exception...
       ListOutParam.Values['КритическаяОшибка']:='Критическая ошибка! Скрипт НЕ
выполнен';
    end;
  END;
end;
```

```
function fsiScript_Run_RT(fsScr: TfsScript;
                          ListScr:TStrings;
                          ListInParam:TStrings;
                          ListOutParam:TStrings;
                          fsCallMethod: TfsCallMethodEvent
                         ):boolean;
//Выполнение FS-скрипта в режиме реального времени
//(при обращении внешнего инициатора информационного обмена)
//  ListScr - текст FS-скрипта
//  ListInParam - Список входных параметров (FS-констант)
//  ListOutParam - Список вЫходных параметров (FS-переменных)
Var
  ItsOk:boolean;
begin
  Result := false;
  ItsOk:=false;
  TRY
    //-----------------------------
    //Инициализация fsScr;
    //конвертация входных и вЫходных параметров FSI;
    //компиляция (контроль ошибок) FS-скрипта
    Result := fsiScript_Compile(fsScr,
                                ListScr,
                                ListInParam,
                                ListOutParam,
                                fsCallMethod
                               );

    //-----------------------------
    if Result then begin
       Result := false;
       //-----------------------------
       //Выполнение FS-скрипта
       fsScr.Execute;
       //-----------------------------
       //-----------------------------
       //Конвертация FS-переменных в выходные параметры FSI
       MyParams_VarFromFS(fsScr, ListOutParam);
       //-----------------------------
       ItsOk:=true;
       Result := true;
    end;
  EXCEPT
    if not ItsOk then begin
       //FS вывалился в Exception...
       ListOutParam.Values['КритическаяОшибка']:='Критическая ошибка! Скрипт НЕ выполнен';
    end;
  END;
end;
```

Рисунок 18 – Функция `fsiScript_Run_RT`