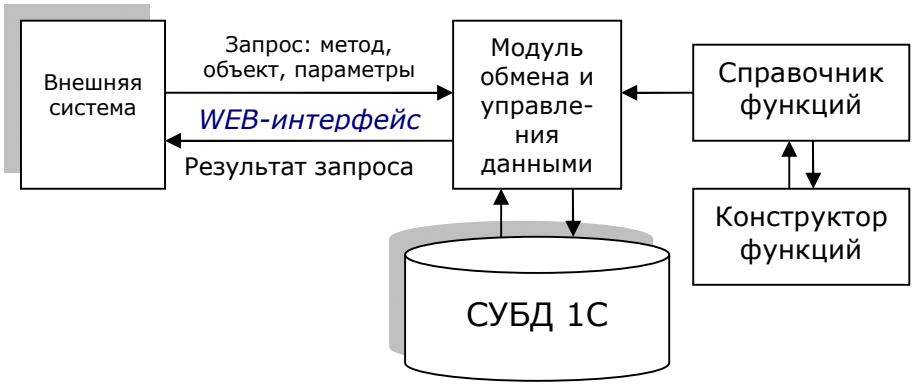
Подсистема Nirax Function

Подсистема "Nirax Function" предназначена для доступа и управления данным 1С из других систем.

Функционально состоит из следующих элементов:
- справочник функций;
- конструктор функций;
- система тестирования функций;
- модуль обмена и управления данными.

Принцип работы следующий.
1. В 1С, с помощью конструктора функций разрабатываются и отлаживаются функции, позволяющие выполнять требуемые действия по доступу и управлению данными. Разработать такую функцию может обычный пользователь, имеющий основные знания по объектам 1С.
2. Из другой системы, через WEB – интерфейс отправляются запросы, содержащие информацию о требуемом действии (методе) над объектом и параметрах действия и объекта.
3. Модуль обмена и управления данными принимает этот запрос, считывает необходимые для выполнения запроса данные из справочника функций и выполняет его.
4. Результат выполнения через WEB – интерфейс возвращается в другую систему.



Обмен и управление данными

Обмен между внешней системой и 1С производится по протоколу SOAP, в котором передается только одна текстовая строка – закодированный по формату JSON оператор запроса.
Для доступа по протоколу SOAP используются следующие данные:
- имя Web-сервиса: NxFunction
- URI пространства имен: http://www.nirax.ru
- имя файла публикации: nxfunction.1cws
- имя метода: NxFunction
- параметры метода – Json /формат string (http://www.w3.org/2001/XMLSchema)/

Внимание!!!
- данные типа дата при кодировании в JSON передаются в следующем виде: "@^ГГГГММДДччммсс", где ГГГГ – год, ММ- месяц, ДД – день, чч – час (от 00 до 23), мм – минуты (от 00 до 59), сс – секунды (от 00 до 59), нулевые значения от конца могут не указываться, например: *"@^20150101"*
- данные типа ссылка при кодировании в JSON передаются в следующем виде: "@^<типБазыДанных>.<имяБазыДанных>?<уникальный идентификатор записи>" например: *:"@^Справочник.Номенклатура?fa268e3f-03b9-11e5-8cfc-002522ee7b83"*

- при передаче строковых данных, начинающихся с "@^" и не являющимися датой или ссылкой – передавать: "@^^".

Оператор запроса

Формат оператора запроса следующий:
{"method":"<имя метода>", "object":<объект>,"parameters":<параметры>}
Возвращает метод данные в следующем виде:
{"Code":<код ошибки>,"Error":"<текст ошибки>","Result":<результат запроса>}
Если запрос выполнен успешно – Code=0, параметр Error – отсутствует, в противном случае Code не равен 0, параметр Error содержит текст ошибки, "Result":nul

В настоящее время работают следующие методы:
GetList – прочитать список функций. Данные об объекте отсутствуют.
Могут быть указаны параметры для отбора:
date – тип: дата - выводить только те функции, дата последнего изменения больше или равна указанной;
verify – тип булево – выводить только проверенные или не проверенные функции – в зависимости от значения параметра: true или false
Метод возвращает таблицу с полями: name – тип строка – имя функции и date – тип дата – дата последнего изменения.

Get – Прочитать – позволяет прочитать данные о функции.
{"method":"Get", "object":{"type":"function","name":"<имя функции>"}
Параметры этим методом не используются.
Возвращает структуру со следующими полями:
- "name":"<имя функции"
-"verify" :<проверено>,
-"date" :"<дата изменения>"
-"modify":"<имя пользователя "
-" help" :"<справка по функции>"
-"parameters": [<массив параметров со следующими полями>]
-"name" :"<имя параметра>"
-"type" :"<тип параметра>"
-"owner":""<владелец параметра> – отражает иерархию
-"help" :"<справка по параметру> ",
-"result":[<массив результатов со следующими полями>]
-"name" :"<имя данных результата>"
-"type" :"<тип данных результата>"
-"owner":""<владелец данных результата> – отражает иерархию
-"help" :"<справка по данному результата> "

Поле тип может принимать следующие значения: string – строка, number – число, boolean – булево, date – дата, structure – структура, table – таблица.

Run – Выполнить – выполняет функцию с заданными параметрами.
{"method":"Run", "object":{"type":"function","name":"<имя функции>"},
"parameters":<параметры>}
Структура параметров и результатов соответствует данным, полученным запросом Get.

Установка Подсистемы

Состав комплекта поставки:
В поставочном варианте идут файлы:
NxFunction_<версия>.cf – файл поставки подсистемы
NxFunction_<версия>.cfu – файл обновления
NxFunctionTest.epf – обработка внешнего тестирования.
readme.txt – инструкция по установке.

Порядок установки
1. Откройте конфигуратор системы, куда будет производиться установка.

Установка производится только в управляемое приложение.
2. Обязательно сделайте копию базы: Администрирование – Выгрузить информационную базу.
3. Начните объединение конфигураций: Конфигуратор – Сравнить, объединить с конфигурацией из файла...
4. Выберите файл поставки: NxFunction_<версия>.cf
После выбора появится окно диалога:
"Обнаружена возможность объединения с постановкой на поддержку.
Конфигурация поставщика:

На Вопрос: "Поставить на поддержку?" ответ – Да.
5. На экране появится таблица сравнения конфигураций.
Снимите отметку со строки свойства. Внимание!!! Если не снимите – придется восстанавливать систему и все повторять с начала.
Нажмите кнопку: Выполнить
6 Появится окно: Настройка правил поддержки. Нажмите Ок.
После этого объединение конфигураций завершено.
7.Если вы объединяете с конфигурацией с основным режимом работы "Объчное приложение", то установите переключатель свойства конфигурации: "Использовать управляемые формы в обычном приложении" в Истина.
8. Для осуществления обмена данными между вашим приложением и внешней подсистемой, конфигурация должна быть обязательно опубликована: Администрирование-Публикация на WEB-сервере.

Порядок обновления:
1. Откройте конфигуратор системы, куда будет производиться установка.
Установка производится только в управляемое приложение.
2. Обязательно сделайте копию базы: Администрирование – Выгрузить информационную базу.
3. Обновите конфигурацию: Конфигурация – Поддержка – Обновить конфигурацию
Появится окно обновления конфигурации. Поставьте отметку на "Выбор файла обновления" и нажмите Далее.
4. Выберите файл обновления: NxFunction_<версия>.cfu
После этого появятся данные о текущей и новой конфигурациях.
Далее появится таблица сравнения конфигураций:
Нажмите кнопку: Выполнить
5 На появившийся вопрос о замещении объектов ответьте "Да".
Появится окно: Настройка правил поддержки. Нажмите Ок.
После этого обновление конфигурации завершено.

Справочник функций

Функции, подготовленные подготовленные для доступа и управления данным 1С хранятся в справочнике функций "NxFunction" и имеет следующие поля и таблицы:
- наименование - может быть любая последовательность букв, цифр и знаков подчеркивания "_", начинающаяся с буквы и не превышающая 50 символов;
- проверена – ставится отметка пользователем, если функция отлажена и проверена ее работоспособность;
- дата изменения – устанавливается автоматически после корректировки;
- изменил – заносится наименование пользователя, который внес последнее изменение;
- справка – пользователь, разработавший функцию вносит справочную информацию по ее использованию;
и таблицы:
- алгоритм – таблица, формируется через конструктор и содержит текст функции;
- поля параметров – таблица, содержит имена, типы, подчиненность и справочную информацию о параметрах функции;
- поля результата – таблица, содержит имена, типы подчиненность и справочную информацию о возвращаемых функцией данных.

Разработка функций производится при помощи конструктора функций.

Конструктор функций.
Формат операторов
Типы данных и операции над ними.
Объекты данных.
Методы

Визуально область формы конструктора разбита на три части.
Левая часть содержит текст функции. Строки текста функции можно набирать непосредственно, особенно при наборе комментариев и вычислений. Но для остальных методов удобней набирать текст на вкладке оператор, расположенной на средней части экрана.
Средняя часть состоит из вкладок:
- оператор – текст текущего оператора функции, представленный в виде дерева;
- параметры – в данную таблицу разработчик вносит данные о полях параметров;
- результаты - в данную таблицу разработчик вносит данные о возвращаемых функцией результатах;
- справка – предназначена для внесения справочной информации о разработанной функции, кроме того содержит: код функции, дату последнего изменения и имя пользователя, выполнившего это изменение;
- программа – служебная вкладка, содержит текст программы на языке 1С и текст выражения, определяющее условие, при котором она выполняется.
Правая часть содержит вкладки со справочной информацией, упрощающую формирование операторов функции:
- методы – перечень методов, доступных для программирования – представлено в виде дерева, подчиненные ветви которого уточняют группу типов объектов (суффикс метода);
- типы – содержит типы объектов, с которыми может работать выбранный метод;
- объекты – содержит перечень имен объектов, выбранного типа;
- поля – содержит перечень имен полей выбранного объекта;
- переменные – содержит перечень имен переменных, определенных в функции.

Формат операторов
В зависимости от используемого метода, может использоваться одна из следующих форматов:
- комментарий, предназначен для комментирования текста программы и отключения от выполнения участков программы при отладке:
// <любой текст>
Примеры:
// Пробел между текстом и двумя наклонными чертами обязателен
// Остаток = Делимое % Делитель

- вычислить, предназначена для выполнения арифметических, логических операций, операций над строками и датами, и стандартных функций над данными:
<имя переменной> = <вычисляемое выражение> <при условии>
Примеры:
Сумма = Слагаемое1 + Слагаемое2
СекундВСутках=60*60*24
Вчера = ТекущаяДата()-СекундВСутках

- методы, содержащие объект непосредственно в операторе (без суффикса, например, метод запрос)
<имя переменной> = <имя метода> <содержимое объекта>, <параметры>, <при условии>
Например:
БанкпГорода = Запрос "ВЫБРАТЬ Банки.Ссылка КАК Ссылка ГДЕ Банки.ЭтоГруппа = ЛОЖЬ И Банки.Город = &Город", ПараметрыЗапроса = Город

- осталные методы:
<имя переменной> = <имя метода> <тип объекта> = <имя объекта>,<параметры>, <при условии>
Пояснения:
- имя переменной - может быть любая последовательность букв, цифр и знаков подчеркивания "_", начинающаяся с буквы, создаваемые имена не должны совпадать с зарезервированными словами, распознавание имен переменных ведется без учета регистра букв, если имя переменной не указано, значение будет присвоено переменной "Результат";

- параметры разделяются запятыми и могут быть:
a) простыми данными: <имя> = <значение>
b) структурой: <имя>=<значение1 ... значение N>
c) таблицей: <имя>=(<значение11 ... значение 1N>,...,<значениеM1 ... значение MN)
В структуре и таблице значения разделяются знаками разделения, определенные методом, и это могут быть знаки: пробел " ", равно "=", точка с запятой ";". В структуре значения можно разделять запятыми, но обязательно тогда заключать в скобки;

- при условии имеет формат: ПриУсловии < выражение>
Выражение аналогично выражению в методе вычислить. Указывается только в тех операторах, которые должны выполняться лишь при определенных условиях, а именно если результат вычисления выражения = ИСТИНА.

Типы данных и операции над ними.

Существую следующие простые типы данных: булево, строка, число, дата, неопределено.

Булево.

Значения данного типа имеют два значения, и представлены следующими литералами: **Истина** и **Ложь**.
Возможны следующие операции: конъюкция "**И**", дизъюнкция "**ИЛИ**", отрицание "**НЕ**". Логические выражения вычисляются слева направо. в следующем порядке: операнды заключенные в скобки, НЕ, И, ИЛИ. Примеры:
ЭтоМужчина = НЕ (ЭтоЖенщина ИЛИ ЭтоРебенок)

Строка

Значения данного типа содержат строку в формате Unicode произвольной длины. Литерал строки представляет собой набор символов заключенных в кавычки. Для задания в строке символа " (кавычка) необходимо записать две кавычки подряд. Например:
Текст = "Эта переменная имеет тип ""Строка""."
Операции над строковыми данными:
- конкатенации (соединения) "+", например: ФИО = Фамилия + " " + Имя + " " + Отчество
- сравнения на: равно "=", больше ">", больше или равно ">=", меньше "<", меньше или равно "<=", не равно "<>" в соответствии с порядком букв в алфавите. Результат сравнения – Истина или Ложь. Например: ЭтоИстина = "Антон" < "Андрей"

Число

Литерал числа представляется набором цифр от 0 до 9. Если число отрицательное, то перед числом может стоять знак "-". Если число дробное, то целая и дробная часть разделяются десятичной точкой ".". Например:
Температура = -10
ПИ = 3.14
Операции:
- сложение "+", вычитание "-", умножение "*", деление "/", остаток от деления "%";
- сравнения на: равно "=", больше ">", больше или равно ">=", меньше "<", меньше или равно "<=", не равно "<>"

Дата

Значения данного типа содержит дату григорианского календаря (с 01 января 0001 года) и время. Литерал заключается в одинарные кавычки и имеет формат: 'ГГГГММДДччммсс', где ГГГГ – год, ММ – месяц, ДД – день месяца, чч – час, мм – минуты, сс – секунды. Допускается опускать последние символы. Примеры: '20150101' - 1 января 2015 года , '20111111231100' – 11 ноября 2011 г. 23:11:00 '00010101' – начало отсчета дат (пустая дата).
Операции:
- сложения "+" и вычитания "-" – к дате можно прибавлять или отнимать только число – определяющее количество секунд;
- сравнения на: равно "=", больше ">", больше или равно ">=", меньше "<", меньше или равно "<=", не равно "<>".

Неопределено
Значение данного типа используется тогда, когда не определен тип значения переменной.
Литерал: Неопределено

Объекты данных.

Представим себе таблицу – каждая таблица содержит колонки, которые мы обычно именуем и будем их называть полями.
Таблица в памяти, которая состоит только из одной строки называется **структурой.**
Пример доступа к полям структуры:

```
Клиент.Имя,
```

где Клиент – имя переменной, содержащей структуру, Имя – наименование поля структуры.
Таблица в памяти, которая может содержать любое количество строк будем называть таблицей значений или просто таблицей. Далее, чтобы не путать строки таблицы с данными типа строка – строки таблицы будем называть записями.
Доступ к записи таблицы производится по ее номеру в квадратных скобках (нумерация начинается от 0).
Например:

```
ФамилияПервогоСотрудника = Сотрудники[0].Фамилия
```

Переменной можно присвоить значение строки таблицы, тогда работа с ней будет аналогична работе со структурой. Пример:

```
Первый Сотрудник= Сотрудники[0]
ФамилияПервогоСотрудника = Первый Сотрудник.Фамилия
```

Таблицы могут располагаться не только в памяти, но и на диске в базах данных.
В базах данных есть таблицы, к записям которых нельзя обращаться по ее адресу. Такие данные будем называть **регистры**. К ним относятся: РегистрыСведений, РегистрыНакопления, РегистрыБухгалтерии и РегистрыРасчета.
Базы данных у которых каждая запись имеет свой адрес будем называть просто **базами данных**, а адрес записи будем называть **ссылка**. К ним относятся следующие типы:
Справочник, Документ, ПланВидовХарактеристик, ПланСчетов, ПланВидовРасчета, БизнесПроцесс, Задача.
В качестве значений полей у базы данных могут быть не только простые типы данных и ссылки, но и таблицы, далее называемые табличные части. Например, в справочнике ФизическиеЛица может находиться поле Дети, которое представляет собой таблицу о детях физического лица.
ПетровАБ = Найти Справочник=ФизическиеЛица, Отбор=(Наименование="Петров А.Б.")
ДетейУПетрова=ПетровАБ.Дети.Количество()
Любая табличная часть имеет поле "НомерСтроки", нумерация производится от 1, т.е. на единицу превышает номер записи в таблице. Адресоваться к строке таблицы можно по номеру записи (нумерация от 0), например:

```
ИмяПервогоРебенкаПетрова= ПетровАБ.Дети[0].Имя
```

Аналогично таблице, переменной можно присваивать значение строки табличной части:

```
ПервыйРебенокПетрова= ПетровАБ.Дети[0]
ИмяПервогоРебенкаПетрова= ПервыйРебенокПетрова.Имя
```

Далее при описании методов объекты данных будем разделять на следующие группы:

Группы	Суффикс
Базы данных	БД
Регистры	РГ
Табличные части (таблицы базы)	ТЧ
Таблицы значений (таблицы)	ТБ
Структуры	СТ

Методы: Комментарий, Вычислить, Завершить, Запрос, Добавить, Найти, Изменить

Ранее рассмотрены методы комментарий и вычислить.
Далее при описании методов мы уберем <при условии>, так как любой метод может его иметь. Во всех методах, если имя переменной не определено – значение присваивается переменной Результат.

Обработка ошибок. Метод Завершить.

При выполнении почти любого метода возможны ошибки. Эти ошибки могут появиться на этапе трансляции, уже после сборки программы непосредственно перед выполнением, и на этапе выполнения. Если ошибки возникли на этапе сборки программы – она просто не выполняется и пользователь информируется об ошибке. Если ошибки возникли на этапе выполнения, то они могут возникнуть как из-за ошибочных исходных данных, вызвавшее прекращение работы метода, так и при не достижении результата. Например, при выполнении таких методов, как Найти – результат не найден. В первом случае программа прекратит свое выполнение, сообщив об ошибке. Во втором случае обработка результата возлагается на пользователя, разрабатывающего функцию. Для возможности завершения функции до достижении ее конца предназначен метод Завершить:
Завершить "<текст ошибки>"
Если текст ошибки="" или "Успешно" - ошибка не произойдет, а результату присвоится значение переменной nx_Результат - Неопределено (null). При необходимости значение переменной можно изменить. Например:

```
Фильтр = Найти Справочник = Номенклатура, Отбор = Артикул = 4477
Завершить "Товар Фильтр не найден" ПриУсловии Фильтр = Неопределено
Завершить "успешно" ПриУсловии Лев(Фильтр.Наименование,6)="Фильтр"
...
```

Запрос.

<имя переменной> = Запрос "<текст запроса>", ПараметрыЗапроса=(<параметры>)
Например:
БанкиГорода = Запрос "ВЫБРАТЬ Банки.Ссылка КАК Ссылка ГДЕ Банки.ЭтоГруппа = ЛОЖЬ И Банки.Город = &Город", ПараметрыЗапроса = Город
Текст запроса обычно формируется через конструктор запроса. Для этого в дереве оператора в поле значения объекта нужно нажать дважды клавишей мыши или кнопку редактирования. При помощи конструктора запроса можно строить сложные запросы, использующие весь спектр таблиц.
Результатом выполнения запроса является таблица значений, с полями, определенными в запросе. Если результат запроса пустой, то и таблица не будет содержать строк.
Для проверки пустого результата можно воспользоваться встроенной функцией Количество. Например:
Завершить "В этом городе нет банков, возможно ошибка в наименовании города", ПриУсловии БанкиГорода.Количество()=0

Далее методы в зависимости от группы объектов будут уточняться суффиксом.

Добавить.

ДобавитьБД – добавить запись в базу данных
<имя переменной> = ДобавитьБД <тип объекта> = <имя объекта>, значения=(<поле> = <значение>, ...), таблицы = (<имя> = <таблица>, ...)

где <поле> - имя поля, которому будет присваиваться <значение>,
<имя> - имя поля табличной части в которую будет загружена таблица значений <таблица>. Метод возвращает ссылку на созданную запись. Например:
Ссылка = ДобавитьБД Справочник = Номенклатура, Значения = (Наименование = "Маслянный фильтр", Артикул = "4477")
!!!! Нет данных если не добавлена

ДобавитьРГ – добавить запись в регистр
<имя переменной> = ДобавитьБД <тип объекта> = <имя объекта>, значения=(<поле> = <значение>, ...), замещать = <замещать>
Аналогично методу ДобавитьБД. Если уже имеется запись с такими ключевыми полями (измерениями) то если Замещать=ЛОЖЬ – замещения записи не произойдет. Во всех остальных случаях произойдет замещение записи.
Возвращает Истина, если запись произведена, ложь - в противном случае. Пример:
Выполнено = ДобавитьРГ Регистрсведений = КурсыВалют, Значения = (Период = Дата, Валюта = Валюта, Курс = Курс, Кратность = 1), Замещать = ИСТИНА

ДобавитьТЧ – добавление записи (строки) в табличную часть.
<имя переменной> = ДобавитьТЧ <тип объекта> = <имя объекта>, ссылка=<ссылка>, значения=(<поле> = <значение>, ...)
где имя объекта составное <имяБазы>.<имяТЧ>, например: ЗаказПокупателя.Запасы;
ссылка- ссылка на запись. Возвращает значение поля добавленной записи: "НомерСтроки". Пример:
Выполнено = ДобавитьТЧ Документ = ЗаказПокупателя.Запасы, Значения = (Номенклатура = Товар.Ссылка, Количество = Количество, Цена = Цена, Сумма = Сумма, СуммаНДС = СуммаНДС, Всего = Всего)

ДобавитьТБ – добавление записи в таблицу значений.
<имя переменной> = ДобавитьТБ Таблица = <имя переменной>, значения=(<поле> = <значение>, ...)
Возвращает индекс созданной строки. Пример:
ИндексСтр = ДобавитьТБ Таблица=Таб, Значения = (Имя = Имя, Количество = Расчет)
!!!!! Пока не реализована.

Найти

НайтиБД – поиск в базе данных
<имя переменной> = НайтиБД <тип объекта> = <имя объекта>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), поля = (<поле, ...)
где условие - знаки операции сравнения: равно "=", больше ">", больше или равно ">=", меньше "<", меньше или равно "<=", не равно "<>".
Параметр отбор определяет по каким полям и значениям производится поиск записи, <направление> – указывает упорядоченность записей по значениям полей при поиске, может иметь значения: УБЫВ или ВОЗР – по умолчанию.
Параметр "поля" какие поля войдут в структуру результата. Если указано одно поле – результат будет значение этого поля, если параметр не указан – возвратится ссылка на найденную запись. Если запись не найдена – возвратится неопределено.

НайтиРГ – поиск в регистре
 <имя переменной> = НайтиРГ <тип объекта> = <имя объекта>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), поля = (<поле, ...)
Абсолютно аналогично НайтиБД. Параметр Поле должен иметь хотябы одно значение. Если запись не найдена – возвратится неопределено.

НайтиТЧ – поиск в регистре
<имя переменной> = НайтиТЧ <тип объекта> = <имя объекта>, ссылка=<ссылка>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), поля = (<поле, ...)

где ссылка – ссылка на запись, в табличной части которой будет производиться поиск. Если параметр поля не указан – возвратится значение поля НомерСтроки.
Если запись не найдена – возвратится неопределено.
Пример:
Цены = НайтиТЧ Документ = ЗаказПокупателя.Запасы, Ссылка = Заказ, Отбор = Номенклатура = Товар.Ссылка, Поля = (Количество, Цена, Сумма)

НайтиТБ – поиск в табличной части
<имя переменной> = НайтиТБ таблица = <имя переменной>, отбор=(<поле> = <значение>, ...), поля = (<поле, ...)
Возвращает - если поля - содержит несколько наименований полей - структуру, если одно поле - то его значение. Если поля не указаны - индекс найденой строки. Если строка не найдена - возвращается: Неопределено.

Изменить

ИзменитьБД – изменить запись в базе данных
<имя переменной> = ИзменитьБД <тип объекта> = <имя объекта>, ссылка = <ссылка>, значения=(<поле> = <значение>, ...)
где ссылка – ссылка не изменяемую запись, значения - поля и значения изменяемые в записи. Возвращает: Истина.

ИзменитьРГ – изменить запись в регистре
<имя переменной> = ИзменитьРГ <тип объекта> = <имя объекта>, отбор=(<поле> = <значение>, ...), значения=(<поле> = <значение>, ...)
где параметр отбор должен содержать все поля типа измерение с их значениями, а значения - поля и значения изменяемые в записи.
Возвращает: Истина

ИзменитьТЧ – изменить запись в табличной части
<имя переменной> = ИзменитьТЧ <тип объекта> = <имя объекта>, ссылка = <ссылка>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), значения=(<поле> = <значение>, ...)
где ссылка – ссылка на запись с изменяемой табличной частью. В качестве отбора для поиска записи рекомендуем использовать значение поля НомерСтроки.
Возвращает: Истина
Пример:
Выполнено = ИзменитьТЧ Документ = ЗаказПокупателя.Запасы, Ссылка = Заказ, Отбор = Номенклатура = Товар.Ссылка, Значения = (Количество = Количество, Цена = Цена, Сумма = Сумма, СуммаНДС = СуммаНДС, Всего = Всего)

ИзменитьТБ – изменить запись в таблице
<имя переменной> = ИзменитьТЧ Таблица = <имя таблицы>, Индекс =<Индекс строки>, значения = (<поле> = <значение>, ...)
Возвращает: Истина.

Выбрать

ВыбратьБД – выбор из базы данных
<имя переменной> = ВыбратьБД <тип объекта> = <имя объекта>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), поля = (<поле, ...)
где условие - знаки операции сравнения: равно "=", больше ">", больше или равно ">=", меньше "<", меньше или равно "<=", не равно "<>".
Параметр отбор определяет по каким полям и значениям производится поиск записи, <направление> – указывает упорядоченность записей по значениям полей при поиске, может иметь значения: УБЫВ или ВОЗР – по умолчанию.
Параметр "поля" какие поля войдут в таблицу резултата, если параметр не указан – то только поле "ссылка". Результат: таблица значений.

ВыбратьРГ – выбор из регистра

<имя переменной> = ВыбратьРГ <тип объекта> = <имя объекта>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), поля = (<поле, ...)
Абсолютно аналогично ВыбратьБД. Параметр Поле должен иметь хотябы одно значение.

ВыбратьТЧ – выбор из табличной части
<имя переменной> = ВыбратьТЧ <тип объекта> = <имя объекта>, ссылка=<ссылка>, отбор=(<поле> <условие> <значение>, ...), порядок = (<поле> <направление>, ...), поля = (<поле, ...)
где ссылка – ссылка на запись, в табличной части которой будет производиться поиск. Параметр отбор определяет по каким полям и значениям производится поиск записи, <направление> – указывает
Параметр "поля" какие поля войдут в таблицу результата, если параметр не указан – то только поле "НомерСтроки". Пример:
Цены = ВыбратьТЧ Документ = ЗаказПокупателя.Запасы, Ссылка = Заказ, Отбор = Номенклатура = Товар.Ссылка, Поля = (Количество, Цена, Сумма)

ВыбратьТБ – выбор из таблицы значений
<имя переменной> = ВыбратьТБ таблица = <имя переменной>, отбор=(<поле> = <значение>, ...), поля = (<поле, ...)
Возвращает - таблицу значений со строками, согласно параметру отбор и полями - в соответствии с параметром поля.

Внимание!!! Планируется расширение количества методов и возможны изменения в уже реализованных.

Система тестирования функций.

Для тестирования разработанных функций, имеются два тестировщика – внутренний и внешний.
Внутренний не требует работы через web-интерфейс.
Внешний – это внешняя обработка, работающая через web-интерфейс. Запускать ее можно в любой конфигурации, кроме испытуемой.
При тестировании нужно указать или выбрать: метод, обычно это Run, объект – это имя функции, и заполнить значения параметров. Нажав кнопку выполнить в поле Запрос будет показан сформированный запрос для выполнения, а в поле Ответ – результат, полученный после выполнения запроса.
На внешнем тестировщике имеется дополнительно страница настройки, где указываются логин, пароль пользователя, для входа в 1С и URL web-интерфейса.
Для удобства имеется переключатель, для изменения URL с внешнего адреса на внутренний и обратно.
В настройках имеется еще поле URI, изменять которое не рекомендуем.

Встроенные функции:
работы со строками, числами, датами, прочие, преобразования, со структурой, с таблицей.

Функции работы со строками:
СтрДлина(<Строка>) – возвращает длину строки.
СокрЛ(<Строка>) – возвращает строку, без начальных пробелов (слева).
СокрП(<Строка>) – возвращает строку, без конечных пробелов (справа).
СокрЛП(<Строка>) – возвращает строку, без начальных и конечных пробелов.
Лев(<Строка>, <Число>) – возвращает строку, содержащую начальные <Число> символов строки.
Прав(<Строка>, <Число>) – возвращает строку, содержащую конечные <Число> символов строки.
Сред(<Строка>, <Начало>, <Число>) – возвращает строку, содержащую конечные <Число> символов строки, начиная с символа <Начало>. Если <Число> не указано – то до конца строки.
Найти(<Строка>, <Подстрока>) – возвращает позицию первого вхождения подстроки в строку, если не находит – 0.
ВРег(<Строка>) – возвращает строку в верхнем регистре.
НРег(<Строка>) – возвращает строку в нижнем регистре.
ТРег(<Строка>) – возвращает строку в которой первая буква каждого слова в верхнем регистре, а остальные в нижнем.
Символ(<КодСимвола>) – возвращает символ, соответствующий коду в формате Unicode.
КодСимвола(<Строка>, <Номер>) – возвращает код символа из позиции <Номер>.
ПустаяСтрока(<Строка>) – возвращает Истина, если в строке нет значащих символов. Не значащие символы это побел, неразрывный пробел (НПП), табуляция (горизонтальная Таб и вертикальная ВТаб), возврат каретки (ВК), перевод строки (ПС), перевод формы (страницы) (ПФ).
Примечание. Спецсимволы можно обозначать: Символы.<спецсимвол), например: *символы.Таб*.
СтрЗаменить(<Строка>, <ПодстрокаПоиска>, <ПодстрокаЗамены>) - находит в исходной строке все вхождения подстроки поиска и заменяет ее на подстроку замены
СтрЧислоСтрок(<Строка>) - позволяет посчитать число строк в многострочной строке. Строки в многострочной строке разделены символами перевода строк (Символы.ПС).
СтрПолучитьСтроку(<Строка>, <Номер>) – получает строку многострочной строки по номеру.
СтрЧислоВхождений(<Строка>, <Подстрока>) - вычисляет число вхождений подстроки в строку.

Функции работы с числами
Цел(<Число>) – целая часть числа
Окр(<Число>, <Разрядность>, <РежимОкругления>) - округляет исходное число до заданной разрядности (по умолчанию 0) в соответствии с заданным режимом округления: РежимОкругления.Окр15как10 или РежимОкругления.Окр15как20 (по умолчанию)
Например, округлим цену до сотен рублей: *ОкругленнаяЦена = Окр(Цена, -2)*
Округлим цену до копеек: *ОкругленнаяЦена = Окр(Цена, 2)*
Log(<Число>) - вычисляет натуральный логарифм параметра <Число>.
Log10(<Число>) - вычисляет десятичный логарифм параметра <Число>.
Sin(<Угол>) - вычисляет синус от аргумента <Угол>, заданного в радианах.
Cos(<Угол>) - вычисляет косинус от аргумента <Угол>, заданного в радианах.
Tan(<Угол>) - вычисляет тангенс от аргумента <Угол>, заданного в радианах.
ASin(<Число>) - вычисляет арксинус от аргумента <Число>.
ACos(<Число>) - вычисляет арккосинус от аргумента <Число>.
ATan(<Число>) - вычисляет арктангенс от аргумента <Число>.
Exp(<Число>) - вычисляет экспоненту (число e) в степени <Число>.
Pow(<ЧислоX>, <ЧислоY>) - возводит число <X> в степень <Y>.
Sqrt(<Число>) - вычисляет квадратный корень параметра <Число>.

Мин(<Значение1>,...) - определяет минимальное значение из полученных параметров.
Макс(<Значение1>,...) - определяет максимальное значение из полученных параметров.
Примечание. В функциях **Мин** и **Макс** значения могут быть типами: число, строка, дата, булево.

Функции работы с датами

ТекущаяДата() - Определяет текущую (системную) дату на компьютере.
Год(<Дата>) - год в указанной дате.
Месяц(<Дата>) – номер месяца в указанной дате.
День(<Дата>) - день месяца в указанной дате.
ДеньНедели(<Дата>) - определяет номер дня недели для указанной даты.
ДеньГода(<Дата>) - определяет номер дня в году для указанной даты.
Час(<Дата>) - час в указанной дате.
Минута(<Дата>) - минута в часе указанной даты.
Секунда(<Дата>) - секунда в минуте в указанной дате.
НачалоГода(<Дата>) - дата и время начала года.
НачалоКвартала(<Дата>) - дата и время начала квартала.
НачалоМесяца(<Дата>) - дата и время начала месяца.
НачалоНедели(<Дата>) - дата и время начала недели.
НачалоДня(<Дата>) - дата и время начала дня.
НачалоЧаса(<Дата>) - дата и время начала часа.
НачалоМинуты(<Дата>) - дата и время начала минуты.
КонецГода(<Дата>) - дата и время конца года.
КонецКвартала(<Дата>) - дата и время конца квартала.
КонецМесяца(<Дата>) - дата и время конца месяца.
КонецНедели(<Дата>) - дата и время конца недели.
КонецДня(<Дата>) - дата и время конца дня.
КонецЧаса(<Дата>) - дата и время конца часа.
КонецМинуты(<Дата>) - дата и время конца минуты.
ДобавитьМесяц(<Дата>, <ЧислоМесяцев>) - Добавляет (или вычитает) к указанной дате заданное число месяцев.

Функции преобразования значений

Булево(<Значение>) - преобразует число в тип Булево: 0 – Ложь, иначе Истина
Число(<Значение>) – преобразует строку или булево в число.
Строка(<Значение>) – преобразует любое значение в строку.
Дата(<"ГГГГММДДччммсс">) – преобразует строку в дату
Дата(<Год>, <Месяц>, <День>, <Час>, <Минута>, <Секунда>) – преобразует числовые значения элементов даты в дату
Тип(<ИмяТипа>) – преобразует строковое имя типа в специальный тип данных: тип. Например: Тип("Строка").
ТипЗнч(<Значение>) – определяет тип значения.
Формат(<Значение>, <ФорматнаяСтрока>) – преобразует значение в строку, полученную в результате форматирования. Форматная строка состоит из параметров форматирования, разделенных точкой с запятой:
ЧЦ (ND) - общее число отображаемых десятичных разрядов целой и дробной частей;
ЧДЦ (NFD) - число десятичных разрядов в дробной части;
ЧН (NZ) - нулевое значение числа., если не задано, то в виде пустой строки, если задано "ЧН=", то в виде "0";
ЧГ (NG) - порядок группировки разрядов числа, по умолчанию обычно 3; чтобы не было группировки: "ЧГ=0";
ДЛФ= Д - дата (цифрами); ДД - длинная дата (месяц прописью); В - полное время, дата может объединяться со временем; ДВ - дата время.
ДФ=<комбинация букв> :
д - день месяца (цифрами) без лидирующего нуля, дд - с лидирующим нулем;
ддд - краткое название дня недели ; дддд - полное название дня недели;
М - номер месяца без лидирующего нуля; ММ - с лидирующим нулем;

ММММ - краткое название месяца; ММММ - полное название месяца;
к - номер квартала в году; г - номер года без века и лидирующего нуля; гг - номер года без века с лидирующим нулем; гггг - номер года с веком;
ч - час в 12 часовом варианте без лидирующих нулей; чч - с лидирующим нулем;
Ч - час в 24 часовом варианте без лидирующих нулей; ЧЧ - с лидирующим нулем;
м - минута без лидирующего нуля; мм - минута с лидирующим нулем;
с - секунда без лидирующего нуля; сс - секунда с лидирующим нулем.

Функции и процедуры работы со структурой

!!! Важно !!! Для выполнения процедур имя переменной должно быть "Процедура".
Новый Структура(<Поля>, <Значение1>, ...) – создает структуру с полями, перечисленными в параметре поля через запятую, и указанными значениями.
например:
Товар = новый структура("Артикул,Наименование","4477","Фильтр маслянный")
<Структура>.**Количество**() - получает количество полей структуры.
<Структура>.**Свойство**(<Поле>) – получает наличие указанного поля в структуре, например: *ЕстьПроизводитель=Товар.Свойство("Производитель")*
Процедура = <Структура>.**Вставить**(<Поле>,<Значение>) – добавляет в структуру поле, например: *Процедура = Товар.Вставить("Производитель","Bosh") При Условии Не ЕстьПроизводитель*
Процедура = <Структура>.**Удалить**(<Поле>) – удаляет из структуры поле.
Процедура = <Структура>.**Очистить**() – удаляет из структуры все поля.

Функции и процедуры работы с таблицами

Количество() - получает количество строк таблицы значений.
Итог(<Поле>) – суммирует значения по полю.