ДОСТИГАЕМ ВМЕСТЕ,
РАЗВИВАЯ КАЖДОГО

Утвержден
БАРМ.00003-39 32 12-ЛУ

Система автоматизации процесса управления государственными и муниципальными закупками – Автоматизированный Центр Контроля – Государственный и муниципальный заказ («АЦК-Госзаказ»/«АЦК-Муниципальный заказ»)

FDS-выражения

Руководство администратора

БАРМ.00003-39 32 12

Листов 37

АННОТАЦИЯ

В документе приводится описание работы подсистемы «FDS-выражения» автоматизированного рабочего места системы «АЦК-Госзаказ»/«АЦК-Муниципальный заказ».

Содержание документа соответствует ГОСТ 19.503-79 «Единая система программной документации. РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА. Требования к содержанию и оформлению».

«Система автоматизации процесса управления государственными закупками - Автоматизированный Центр Контроля – Государственный заказ» («АЦК-Госзаказ») зарегистрирована в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам, Свидетельство № 2008610925 от 21 февраля 2008 г. «Система автоматизации процесса управления муниципальными закупками - Автоматизированный Центр Контроля - Муниципальный заказ» («АЦК-Муниципальный заказ») зарегистрирована в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам, Свидетельство № 2009615485 от 02 октября 2009 г.

ООО «БФТ» оставляет за собой право вносить изменения в программное обеспечение без внесения изменений в эксплуатационную документацию.

Оперативное внесение изменений в программное обеспечение отражается в сопроводительной документации к выпускаемой версии.

Документ соответствует версии системы ««АЦК-Госзаказ»/«АЦК-Муниципальный заказ»» – 1.39.0. Последние изменения внесены 27.06.2019 г.

СОДЕРЖАНИЕ

1

Общие положения

FDS-выражения представляют собой механизм использующийся в описаниях метаданных различных настраиваемых компонентов, проверок на выполнение тех или иных условий и ряде других ситуаций.

В зависимости от того, где происходит интерпретация **FDS-выражения**, его содержимое транслируется либо в **SQL-выражение** (на сервере), либо в **FBL-скрипт** (на клиенте).

Данный документ содержит описание языка **FDS-выражений**, его синтаксиса и семантики.

2

Назначение механизма

FDS-выражения используются в описании метаданных следующих настраиваемых компонентов:

Таблица 1 – Компоненты в которых используются FDS-выражения

№ п/п	Компонент	Где используется
1	Объект данных	Описание связи с другими объектами данных в виде выражения (через соотношения между полями объектов, которые не могут быть описаны через ссылки).
2	Серверный валидатор объекта данных	Выражение проверки и выражение сообщения об ошибке.
3	FDS-провайдер	Описание колонки провайдера, описание заголовка (header) провайдера, описание фильтра провайдера, описание возвращаемого агрегированного значения.
4	Генератор документов	Описание фильтра для выборки документов-источников, описание фильтра для выборки набора строк из многострочных документов.
5	Контур проверки ВКМ	Выражение проверки и выражение сообщения об ошибке.
6	Механизм проверки документов на статусах	Описание условий, выполнение которых требуется для инициализации того или иного действия, привязанного к конкретному статусу документа.
7	Механизм фильтров по умолчанию	Вычисление для фильтров значений по умолчанию.

3

Термины и определения

Агрегация – процесс и результат выборки полей одного или нескольких объектов и объединения их в рамках одного метаобъекта для дальнейшего оперирования его данными.

Метаобъект – абстрактный предметный объект, образующийся из полей другого объекта (объектов) путем агрегации.

Контекст интерпретации – программное окружение, в пределах которого выполняется **FDS-выражение**.

Lookuper – серверный класс, позволяющий управлять отношениями (связями) между объектами с помощью их метаописаний.

Контур проверок – совокупность атомарных проверок, каждая из которых представляет собой отдельное правило валидации объекта.

DocRetention – механизм, автоматизирующий задачу контроля сроков пребывания электронного документа на статусах.

StateTranslator – класс, являющийся обработчиком любых изменений элементов управления клиентской формы документа.

DocumentValidator – класс, выполняющий проверку заполненности и валидности данных полей клиентской формы при инициализации события **Сохранить**.

4

Описание синтаксиса

4.1 Типы данных

FDS-выражения работают с данными следующих типов:

Таблица 2 – Типы данных, работающие с FDS-выражениями

Логический тип АЦК	Описание	Параметры	Назначение
LONG	8-ми байтовое целое.	Нет	Ссылки, связи (**ID**), целочисленные данные.
CURRENCY	Вещественное, до 4-х знаков после запятой.	Нет	Числовые показатели.
STRING	Строка символов.	Длина (количество символов)	Строковые константы и переменные.
DATETIME	Дата и время.	Нет	Дата и время.
BCD	Вещественное с заданным количеством знаков после запятой.	Точность (количество знаков после запятой)	Коэффициенты, требующие высокой точности.
BINARY	Двоичные данные.	Нет	Файлы, картинки, аттачи.

4.2 Выражения

Основой языка **FDS-выражений** является **выражение** (**FDS-выражение**).

FDS-выражение – это конструкция языка **FDS-выражений**, оперирующая определенными данными (операндами), преобразующая их с помощью специальных элементов языка (операторов) и возвращающая результат определенного типа.

В **FDS-выражениях** используются следующие операнды:

- поля объектов;
- параметры;
- функции;
- константы.

Для манипулирования данными и управления вычислительным процессом в **FDS-выражениях** используются следующие операторы:

- арифметические;
- логические;

- операторы сравнения;
- условные.

В простейшем случае **FDS-выражение** может состоять из одного операнда.

Синтаксис простого **FDS-выражения**:

{поле объекта} | {:параметр} | Функция() | константа

Пример. *Простое FDS-выражение:* ***{DATE}***

Синтаксис составного **FDS-выражения**:

(выражение)

| +|- выражение

*| выражение *\|/\|+|- выражение*

Пример. *Составное FDS-выражение:* ***{QUANTITY} + 5***

4.3 Операнды

4.3.1 Поля объектов

Часто на этапе составления **FDS-выражений** нет возможности указать фактическое описание поля в контексте исполняемого **SQL-выражения** в целом. Такие поля в выражениях описываются с помощью фигурных скобок в формате:

{<полное_имя_поля>}

Вследствие того, что **FDS-выражение** может содержать поля из разных взаимосвязанных объектов (таблиц), принадлежность поля к тому или иному объекту определяется с помощью **полного_имени_поля**. Полное имя поля представляет собой комбинацию **пути_к_полю и имени_поля**. В качестве разделителя элементов полного имени поля используется символ «.» (точка).

{<путь_к_полю>.<имя_поля>}

Путь_к_полю – это выражение, определяющее, к какому объекту относится описываемое поле. Существует несколько способов описания принадлежности поля какому-либо объекту в рамках **FDS-выражения**:

1. С помощью цепочки связей от мастер-объекта компонента к объекту, содержащему описываемое поле.

Мастер-объектом компонента называется объект, от которого по описанным в системе связям объектов, входящих в **FDS-выражение**, можно дойти до любого из них. Мастер-объект задается в контексте описания компонента, в котором используется данное выражение. Для полей мастер-объекта **путь_к_полю** равен пустой строке. Для всех других объектов он равен набору логических имен связей, разделенных между собой символом «.» (точка) от мастер-объекта к объекту, содержащему поле.

Таблица 3 – Пример описания пути к полю через цепочку связей

Выражение	Описание
{NAME}	Поле **NAME** мастер-объекта.
{M_to_Obj1.NAME}	Поле **NAME** объекта **Obj1**, с которым мастер-объект связан внешним ключом **M_eo_Obj1**. Данный внешний ключ принадлежит мастер-объекту.
{M_to_Obj1.Obj1_to_Obj2.NAME}	Поле **NAME** объекта **Obj2**, с которым связан объект **Obj1** посредством своего внешнего ключа **Obj1_to_Obj2**. Объект **Obj1**, в свою очередь, связан с мастер-объектом с помощью внешнего ключа мастер-объекта **M_to_Obj1**.

2. С помощью предопределенных префиксов **this** и **foreign**.

Такой способ используется при описании связей между двумя объектами через выражение. Соответственно, префикс **this** соотносится с объектом, которому принадлежит описываемый внешний ключ, а префикс **foreign** с объектом, связь к которому описывается.

Таблица 4 – Пример описания пути к полю через префиксы «this» и «foreign»

Выражение	Описание
{this.NAME} = {foreign.NAME}	Связь (внешний ключ) объекта, с которым ассоциирован префикс **this** с объектом, с которым ассоциирован префикс **foreign**, через равенство значений полей **NAME**.
{this.NAME} = {foreign.NAME} and {this.CODE} = {foreign.CODE}	Аналогично выше приведенному выражению, только связь строится через равенство значений полей **NAME** и **CODE**.

3. С помощью определяемых разработчиком префиксов.

Определение разработчиком префиксов представляет собой детализацию **FDS-выражения** и заключается в явном указании объекта данных, ассоциированного с каждым из префиксов, а также пути от указанных объектов к мастер-объекту выражения (не путать с мастер-объектом компонента). Мастер-объектом выражения является объект, к которому можно добраться от любого объекта

данных, участвующего в выражении, через описанные в системе связи этих объектов. Путь к мастер-объекту описывается в виде цепочки имен связей соответствующих объектов. Для мастер-объекта путь не заполняется.

Примечание.

- *Этот способ используется в случаях, когда в контексте компонента, содержащего данное выражение, не определен мастер-объект. Также допустимо использовать данный способ указания **пути_к_полю** в **FDS-выражениях** компонентов, в контексте которых мастер-объект определен. В этом случае мастер-объект выражения должен совпадать с мастер-объектом компонента.*

- *Поля объектов, описанных в детализации и не являющихся мастер-объектом, должны быть аргументами агрегирующих функций.*

- *При именовании префиксов можно использовать только латинские символы.*

Таблица 5 – Пример описания пути к полю через определяемые разработчиком префиксы

Выражение	Детализация	Описание
{obj1.AMOUNT} > SUM({obj2.AMOUNT})	объекты: **object1**, **object2**. Префиксы: **obj1**, **obj2**. Путь к мастер-объекту: **obj2_to_obj1**.	В выражении участвует два объекта: **object1** и **object2**, для которых определены соответствующие префиксы **obj1** и **obj2**.

4.3.2 Параметры

В **FDS-выражениях** используются именованные параметры. Имя параметра должно быть заключено в фигурные скобки с двоеточием в соответствии со следующим форматом: *{:<имя_параметра>}*.

Параметры делятся на:

- **сессионные** – параметры, зависящие от сессии и вычисляемые системой, например:
  - **{:SESSION_USERID}** – ID пользователя;
  - **{:SESSION_SESSIONID}** – ID сессии;
  - **{:NOW}** – текущее (системное) время и дата.

- **системные** – параметры, хранящиеся в таблице **SYSPARAM** базы данных, например:
  - **{:client_resource_description}** – описание клиентских ресурсов;
  - **{:client_update_type_default}** – тип обновления по умолчанию;
  - **{:font_styles}** – стиль шрифта.

4.3.3 Функции

В **FDS-выражениях** возможно использование функций. В качестве аргументов функций могут использоваться поля объектов, **FDS-параметры** и другие функции.

Общие синтаксис функций имеет вид:

ИмяФункции(<аргумент>[, <аргумент>, <аргумент>, ...])

Пример: *DateDiff({FIRST_DATE}, {LAST_DATE}, DAY)*

Синтаксис всех функций **FDS-выражений** чувствителен к регистру. Аргументами функций могут являться поля объектов, **FDS-параметры**, константы и другие функции.

4.3.3.1 Функции работы с датами

В **FDS**-выражениях используются следующие функции работы с датами.

4.3.3.1.1 TruncateTime

ФУНКЦИЯ ОБНУЛЕНИЯ ВРЕМЕНИ В ПОЛНОМ ФОРМАТЕ ДАТЫ.

Синтаксис:

TruncateTime(<полная дата>: datetime): datetime

Пример: *TruncateTime({DATE})*

4.3.3.1.2 DateDiff

ВЫЧИСЛЕНИЯ РАЗНИЦЫ МЕЖДУ ДВУМЯ ДАТАМИ.

Синтаксис:

DateDiff(<начальная дата>: datetime, <конечная дата>: datetime, <единица измерения>[, <список id>: string]): currency

В качестве параметра <*единица измерения*> используется одно из следующих зарезервированных слов:

- *SECOND* (секунда);
- *MINUTE* (минута);
- *HOUR* (час);
- *DAY* (день).

Параметр <*список id*> – перечень уникальных идентификаторов типа дня из таблицы *CLNDDAYTYPE* базы данных (перечисленных через запятую или запятую и пробел). Данный параметр используется, если <*единица измерения*> = *DAY.* В случае, если <*единица измерения*> = *DAY* и <*список id*> опущен, то за единицу измерения принимается календарный день. Границы интервала включаются в вычисляемую разницу.

Пример: *DateDiff({FIRST_DATE}, {LAST_DATE}, DAY, '360038')*

4.3.3.1.3 DateAdd

ФУНКЦИЯ ДОБАВЛЕНИЯ ИНТЕРВАЛА (числового значения) К ДАТЕ.

Синтаксис:

DateAdd(<начальная дата>: datetime, <интервал>: long, <единица измерения>[, <список id>: string]): datetime

В качестве параметра <единица измерения> используется одно из следующих зарезервированных слов:

- *SECOND* (секунда);
- *MINUTE* (минута);
- *HOUR* (час);
- *DAY* (день);
- *WEEK* (неделя).

Параметр <*список id*> – перечень уникальных идентификаторов типа дня из таблицы *CLNDDAYTYPE* базы данных (перечисленных через запятую или запятую и

пробел). Данный параметр используется, если <*единица измерения*> = *DAY*. В случае, если <*единица измерения*> = *DAY* и <*список id*> опущен, то за единицу измерения принимается календарный день.

Пример: *DateAdd({FIRST_DATE}, 10, DAY, '360038')*

4.3.3.1.4 DatePart

ФУНКЦИЯ ИЗВЛЕЧЕНИЯ ЧАСТИ ДАТЫ (года, месяца, дня, часа, минут, секунд) В ВИДЕ ЦЕЛОГО ЧИСЛА.

Синтаксис:

DatePart(<полная дата>: datetime, <единица измерения>): long

В качестве параметра <*единица измерения*> используется одно из следующих зарезервированных слов:

- *SECOND* (секунда);
- *MINUTE* (минута);
- *HOUR* (час);
- *DAY* (день);
- *MONTH* (месяц);
- *YEAR* (год).

Пример: *DatePart({MY_DATE}, YEAR)*

4.3.3.1.5 GetPhaseDate

Синтаксис:

Date(Long doc_id, Long code, Timestamp default)

Возвращает дату наступления фазы **code** для решения **id**, либо дату по умолчанию **default**, транслируя параметры в **SQL-выражение** вида:

coalesce((select phasedate from cmpphase where order_id = doc_id and phasecode = code), default)

Пример: *DateDiff(GetPhaseDate({PARENT.CMPORDER.ID}, 1, {:NOW}), {:NOW}, DAY, '1') - 2 > StrToLong({:order_compete_selection_result_notice_term})*

4.3.3.1.6 GetNoticeDocDate

Синтаксис:

Timestamp GetNoticeDocDate(Long doc_id, Long group_id, Long is_pub, Long status_id)

Возвращает минимальную дату всех извещений документа **doc_id** группы **group_id** на статусе **status_id** и имеющих признак публичного доступа **is_pub**. Формирует **SQL-запрос** вида:

select min(n.doc_date) from noticedoc n join document nd on n.document_id=nd.id where nd.parent_id=doc_id and n.sysdocgroup_id=group_id and n.ispublic=is_pub and n.dispstatus_id=status_id

Пример: *{DOC_DATE} = GetNoticeDocDate({ID}, 5, 0, 0)*

4.3.3.2 Функции приведения типов

Таблица 6 – Функции приведения типов в FDS-выражениях

	LONG	CURRENCY	BCD	STRING	DATETIME	BINARY
LONG		LongToCur()	LongToBCD()	LongToStr()		
CURRENCY	CurToLong()			CurToStr()		
BCD	BCDToLong()			BCDToStr()		
STRING	StrToLong()	StrToCur()	StrToBCD()		StrToDate()	
DATETIME				DateToStr()		
BINARY						

В операциях с типами **Currency** и **BCD** производится неявное преобразование **Currency** в **BCD**.

4.3.3.2.1 CurToStr

ВЕЩЕСТВЕННОЕ В СТРОКУ.

Синтаксис:

CurToStr(<вещественное значение>: currency, <количество знаков после запятой>: long[, <разделитель групп разрядов для целой части>: string]): string

Разделитель целой и дробной части «.» (точка). Если параметр *<разделитель групп разрядов для целой части>* не указан, то возвращается строка без разделителей групп разрядов.

Пример: *CurToStr({MY_CUR_VALUE}, 2, ' ')*

4.3.3.2.2 StrToLong

СТРОКА В ЦЕЛОЕ.

Синтаксис:

StrToLong(<строка>: string[, <разделитель групп разрядов>: string]): long

Если указан параметр *<разделитель групп разрядов>* при преобразовании происходит **trim** символов *<разделитель групп разрядов>.*

Пример: *StrToLong({MY_STR}, ' ')*

4.3.3.2.3 StrToCur

СТРОКА В ВЕЩЕСТВЕННОЕ.

Синтаксис:

StrToCur(<строка>: string[, <разделитель групп разрядов>: string]): currency

Функция принимает строку с разделителями целой и дробной части: точкой (.) и запятой (,). Если указан параметр *<разделитель групп разрядов>,* то при преобразовании происходит **trim** символов *<разделитель групп разрядов>*.

Пример: *StrToCur({MY_STR}, ' ')*

4.3.3.2.4 StrToDate

СТРОКА В ДАТУ.

Синтаксис:

StrToDate(<строка>: string): datetime

Функция возвращает:

- дату формата *dd.mm.yyyy hh:nn:ss.z*, если принимает строку формата *dd.mm.yyyy hh:nn:ss.z*;
- дату формата *dd.mm.yyyy hh:nn:ss*, если принимает строку формата *dd.mm.yyyy hh:nn:ss*;
- дату формата *dd.mm.yyyy 00:00c00*, если принимает строку формата *dd.mm.yyyy*.

Пример: *StrToDate({MY_STR})*

4.3.3.2.5 DateToStr

ДАТА В СТРОКУ.

Синтаксис:

DateToStr(<дата>: datetime[, <формат даты/времени>: string]): string

В качестве параметра *<формат даты/времени>* используется строка, составленная с помощью описателей.

Таблица 7 – Описатели строки

Описатель	Отображает
d	День в виде числа без ведущего нуля.
dd	День в виде числа с ведущим нулем.
ddd	День недели в виде аббревиатуры из трех букв (см. Приложение 1).
dddd	День недели в виде полного наименования (см. Приложение 1).
m	Месяц в виде числа без ведущего нуля либо минуты, если следует за **h** или **hh**.
mm	Месяц в виде числа с ведущим нулем либо минуты, если следует за **h** или **hh**.
mmm	Название месяца в виде аббревиатуры из трех букв (см. Приложение 1).
mmmm	Полное название месяца (см. Приложение 1).
yy	Год посредством двух младших разрядов.
yyyy	Год в полном, четырехразрядном формате.
h	Час без ведущего нуля.
hh	Час с ведущим нулем.
n	Минуты без ведущего нуля.
nn	Минуты с ведущим нулем.
s	Секунды без ведущего нуля.

ss	Секунды с ведущим нулем.
z	Миллисекунды.

Используемый разделители для дат «.» (точка), для времени «:» (двоеточие). Если формат даты не указан, то возвращается строка вида: *dd.mm.yyyy hh:nn:ss*.

Пример: *DateToStr({MY_DATE}, 'dd.mm.yyyy')*

4.3.3.2.6 Coalesce

Синтаксис:

Timestamp Coalesce(Timestamp param1, Timestamp param2)

Long Coalesce(Long param1, Long param2)

String Coalesce(String param1, String param2)

Boolean Coalesce(Boolean param1, Boolean param2)

Транслирует **param1** и **param2** в **SQL-выражение** вида:

- для параметров типа отличного от **Timestamp**:

coalesce(param1, param2)

- для **firebird** (параметры типа **Timestamp**):

coalesce(cast(param1 as timestamp), cast(param2 as timestamp))

- для **oracle** (параметры типа **Timestamp**):

coalesce(cast(param1 as date), cast(param2 as date))

Пример: *DateDiff(Coalesce({PARENT.CMPORDER.REFUSE_DATE}, {:NOW}), {:NOW}, DAY, '1') - 2 > StrToLong({:order_compete_open_declinenotice_term})*

4.3.3.2.7 IsNull

Синтаксис:

Boolean IsNull(Timestamp param1)

Boolean IsNull(Long param1)

Boolean IsNull(String param1)

Boolean IsNull(Boolean param1)

Возвращает переданный параметр **NULL**. Транслирует **param1 SQL-выражение** вида:

param1 is null

Пример: *IsNull({STATECONTRACT.PROCESS_BEFORE_DATE})*

4.3.3.3 Функции работы с документами

4.3.3.3.1 HasPubNoticeDoc

Синтаксис:

Boolean HasPubNoticeDoc(Long doc_id, Long group_id, Long is_pub, Long status_id)

Проверяет у документа **doc_id** наличие извещений группы **group_id** на статусе **status_id** и имеющих признак публичного доступа **is_pub**. Формирует **SQL-запрос** вида:

exists (select null from noticedoc n join document nd on n.document_id=nd.id where nd.parent_id=doc_id and n.sysdocgroup_id=group_id and n.ispublic=is_pub and n.dispstatus_id=status_id)

Пример: *DateDiff(GetPhaseDate({PARENT.CMPORDER.ID}, 1, {:NOW}), {:NOW}, DAY, '1') - 2 > StrToLong({:order_compete_selection_result_notice_term})) AND (HasPubNoticeDoc({ID}, 7, 0, 0)*

4.3.3.3.2 HasAnyNoticeDoc

Синтаксис:

Boolean HasAnyNoticeDoc(Long doc_id, Long group_id, Long status_id, const Long invert_status)

Проверяет у документа **doc_id** наличие извещений группы **group_id** на статусе **status_id** (если **invert_status=0**), либо на статусах отличных от **status_id** (если **invert_status=1**) без учета признака публичного доступа.

В случае параметра **invert_status=0** формирует **SQL-запрос** вида:

exists (select null from noticedoc n join document nd on n.document_id=nd.id where nd.parent_id=doc_id and n.sysdocgroup_id=group_id and n.dispstatus_id = status_id)

В случае параметра **invert_status=1** формирует **SQL-запрос** вида:

exists (select null from noticedoc n join document nd on n.document_id=nd.id where nd.parent_id=doc_id and n.sysdocgroup_id=group_id and n.dispstatus_id <> status_id)

Пример: *DateDiff(Coalesce({CMPORDER.OPENINGDATE}, {:NOW}), {:NOW}, DAY, '1') - 2 > StrToLong({:order_publish_contest_period})) AND (NOT HasAnyNoticeDoc({ID}, 5, 10, 0))*

4.3.3.3.3 IsDocFlagSet

Синтаксис:

Boolean IsDocFlagSet(Long doc_id, Long flag_id)

Проверяет у документа **doc_id** наличие установленного флага документа **flag_id** формируя **SQL-запрос** вида:

exists (select null from docflag where document_id=doc_id and docflagtype_id=flag_id)

Пример: *({NOTICE.SYSDOCGROUP_ID}=5) AND ({PARENT.CMPORDER.PURCHASEMODE_ID}=1) AND (IsDocFlagSet({PARENT.CMPORDER.DOCUMENT_ID}, 14))*

4.3.4 Константы

В **FDS**-выражениях могут использоваться строковые и числовые константы:

Пример. *Строковая константа: 'Ошибка!'.*

Числовая константа: 25.

4.4 Операторы

4.4.1 Арифметические операторы

Таблица 8 – Арифметические операторы, используемые в FDS-выражениях

Оператор	Описание
+	Сложение, Конкатенация
-	Вычитание
*	Умножение
/	Деление

Таблица 9 – Возможность межтипового использования арифметических операторов и типы результата операций (на данный момент)

	LONG	CURRENCY	BCD	STRING	DATETIME	BINARY
LONG	+, -, *, LONG; / - CURRENCY	+, -, *, /, CURRENCY	+, -, *, /, BCD	-	+, -, DATETIME	-
CURRENCY	+, -, *, /, CURRENCY	+, -, *, /, CURRENCY	+, -, *, /, BCD	-	+, -, DATETIME	-
BCD	+, -, *, /, BCD	+, -, *, /, BCD	+, -, *, /, BCD	-	+, -, DATETIME	-
STRING	-	-	-	+, STRING	-	-
DATETIME	+, -, DATETIME	+, -, DATETIME	+, -, DATETIME	-	-, CURRENCY	-
BINARY	-	-	-	-	-	-

Таблица 10 – Возможность межтипового использования арифметических операторов и типы результата операций (в перспективе):

	LONG	CURRENCY	BCD	STRING	DATETIME	BINARY
LONG	+, -, *, LONG; / - CURRENCY	*, /, CURRENCY	*, /, BCD	-	-	-
CURRENCY	*, /, CURRENCY	+, -, *, /, CURRENCY	+, -, *, /, BCD	-	-	-
BCD	*, /, BCD	+, -, *, /, BCD	+, -, *, /, BCD	-	-	-
STRING	-	-	-	+, STRING	-	-
DATETIME	-	-	-	-	-	-
BINARY	-	-	-	-	-	-

4.4.2 Логические операторы

Таблица 11 – Логические операторы, используемые в FDS-выражениях

Оператор	Описание
AND	Логическое **И**
OR	Логическое **ИЛИ**
NOT	Логическое отрицание (**НЕ**)

4.4.3 Операторы сравнения

Таблица 12 – Операторы сравнения, используемые в FDS-выражениях

Оператор	Описание
=	Равно
>	Больше
<	Меньше
>=	Больше либо равно
<=	Меньше либо равно
<>	Нв равно

Таблица 13 – Возможность межтипового использования операторов сравнения (на данный момент)

	LONG	CURRENCY	STRING	DATETIME	BCD	BINARY
LONG	=, >, <, >=, <=, <>	=, >, <, >=, <=, <>	-	-	-	-
CURRENCY	=, >, <, >=, <=, <>	=, >, <, >=, <=, <>	-	-	-	-
STRING	-	-	=, >, <, >=, <=, <>	-	-	-
DATETIME	-	-	-	=, >, <, >=, <=, <>	-	-
BCD	-	-	-	-	-	-
BINARY	-	-	-	-	-	-

Таблица 14 – Возможность межтипового использования операторов сравнения (в перспективе)

	LONG	CURRENCY	BCD	STRING	DATETIME	BINARY

LONG	=, >, <, >=, <=, <>	-	-	-	-	-
CURRENCY	-	=, >, <, >=, <=, <>	=, >, <, >=, <=, <>	-	-	-
BCD	-	=, >, <, >=, <=, <>	=, >, <, >=, <=, <>	-	-	-
STRING	-	-	-	=, >, <, >=, <=, <>	-	-
DATETIME	-	-	-	-	=, >, <, >=, <=, <>	-
BINARY	-	-	-	-	-	-

4.4.4 Условные операторы

Синтаксис:

```
CASE
WHEN <условие_1> THEN <выражение_1>
[WHEN <условие_2> THEN <выражение_2>
...
WHEN <условие_m> THEN <выражение_m>]
[ELSE <выражение_n>]
END
```

Где *<условие_1>*, *<условие_2>*, *<условие_m>* являются логическими операциями, а *<выражение_1>*, *<выражение_2>*, *<выражение_n>* должны быть одинакового типа.

Пример: *CASE*

```
WHEN {IS_CFO} = 1
THEN {QUANTITY} + 5
ELSE {QUANTITY} - 5
END
```

5

Принципы функционирования

Основной особенностью **FDS-выражений** является то, что правила их составления (агрегации) и интерпретации различаются в зависимости от того, где и для каких целей (в каком контексте) используется выражение.

5.1 Контексты интерпретации

Таблица 15 – Контексты интерпретации FDS-выратений

Контекст	Где выполняется	Описание
Lookuper	Серверная часть	В данном контексте **FDS-выражения** используются при работе с: • lookup-полями объектов данных; • refpath'ами при создании метаобъектов; • FDS-провайдерами (колонки, фильтры, заголовки).
Контуры проверок	Серверная часть	В данном контексте **FDS-выражения** используются при составлении условий атомарных проверок.
DocRetention	Серверная часть	В данном контексте **FDS-выражения** используются для описания условий, выполнение которых требуется для инициализации того или иного действия, привязанного к конкретному статусу документа.
StateTranslator	Клиентская часть	В данном контексте **FDS-выражения** используются для составления условий проверки и установки значений свойств элементов управления клиентской формы.
DocumentValidator	Клиентская часть	В данном контексте **FDS-выражения** используются для составления условий проверки заполненности и валидности данных полей клиентской формы.

5.2 Правила интерпретации

5.2.1 Lookuper

В данном контексте **FDS-выражение** транслируется в язык **SQL**. При интерпретации **FDS-выражения Lookuper** происходит замена полных имен полей с путями к ним фактическими наименованиями полей, а параметров их фактическими значениями.

FDS-выражение:

{IS_CFO}={:IS_CFO}

SQL-выражение:

SELECT <колонки> FROM <мастер-объект> WHERE IS_CFO=<значение параметра IS_CFO>

Особенности интерпретации **FDS-выражений** в провайдерах:

- если атрибут колонки **ISGROUP=1**, то к остальным колонкам, у которых **ISGROUP=0**, добавляется **SUM()**;
- нельзя использовать агрегирующие функции в описаниях колонок **FDS-провайдеров**.

5.2.2 Контуры проверок

В данном контексте **FDS-выражение** транслируется в язык **SQL**. При интерпретации **FDS-выражения** в контурах проверок происходит замена полных имен полей с путями к ним фактическими наименованиями полей, а параметров их фактическими значениями. Кроме того, происходит проверка **FDS-выражения** и по ее результатам возвращается **true** (1) или **false** (0), которые и являются результатами атомарной проверки.

FDS-выражение:

```
({o.IS_CFO} = 1) or ({o.IS_FIRM} = 1) or ({o.IS_DEP} = 1)
```

SQL-выражение:

```
SELECT
case when ((o.IS_CFO = 1) or  (o.IS_FIRM = 1) or (o.IS_DEP = 1)) then 1 else 0 end
FROM ORG o
WHERE <название ключевого поля> = <значение ключевого поля>
```

5.2.3 DocRetention

В данном контексте **FDS-выражение** транслируется в язык **SQL**. При интерпретации **FDS**-выражения механизмом **DocRetention** происходит замена полных имен полей с путями к ним фактическими наименованиями полей, а параметров их фактическими значениями. Результатом выполнения выражения являются записи, удовлетворяющие условию выражения, к которым впоследствии будут применены правила оформления, определенные в таблице **DOCRETENTION**.

FDS-выражение:

({:NOW} - {SYSTEM_DATE} < 5) AND ({MY_ FIELD} > {:MY_SYSPARAM})

SQL-выражение:

SELECT DISTINCT M.ID DOCUMENT_ID, DR.ID DOCRETENTION_ID, DR.FONT, M.DOCUMENTCLASS_ID, M.INSTANCE_LINK, DOC_NUMBER MSG

FROM DOCRETENTION DR

JOIN DOCSTATUS DS ON DR.DOCSTATUS_ID = DS.ID

JOIN DOCUMENT M ON M.DOCUMENTCLASS_ID=DS.DOCUMENTCLASS_ID AND M.DISPSTATUS_ID=DS.DISPSTATUS_ID

WHERE (<системная дата> - <значение поля SYSTEM_DATE> < 5) AND (<значение поля MY_FIELD> > <значение параметра MY_SYSPARAM>)

AND DR.ID=<ID правила оформления>

Правилами оформления является ряд настроек визуального оповещения пользователя.

5.2.4 StateTranslator

В данном контексте **FDS-выражение** транслируется в язык **FBL**. Результат **FDS-выражения** присваивается значению свойства того или иного элемента управления формы.

FDS-выражение (в составе XML):

<STATE CONTROL= "edtCHILD_ID1" PROPERTY= "VISIBLE" STATE= "({OBJECTKIND_ID} > = 21) AND ({OBJECTKIND_ID} <= 26)"/>

FBL-выражение:

edtCHILD_ID1YVISIBLE := ((OBJECTKIND_ID >= 21) and (OBJECTKIND_ID <= 26)

В данном примере значению свойства **visible** элемента управления **edtCHILD_ID1** присваивается значение **true** или **false**, в зависимости от результата вычисления выражения.

Также в **StateTranslator** реализован механизм **Lookuper**, позволяющий получать доступ к атрибутам объекта через префиксы и отношения (связи) между объектами.

FDS-выражение (в составе XML):

<STATE CONTROL= "edtCHILD_ID1" PROPERTY= "TEXT" STATE= "{obj1.NAME}" />

FBL-выражение:

```
edtCHILD_ID1.TEXT := ResolvePath(MetaObjectName, Data, 'obj1.NAME');
```

Использующаяся в этих случаях функция **ResolvePath** принимает следующие аргументы:

- **MetaObjecиName** – название корневого метаобъекта (можно использовать предопределенную в модуле **Lookuper** константу **MetaObjectName** или прямо (по имени) указать в вызове константу, соответствующую метаобъекту);
- **Data** – переменная, определенная в модуле **NXML** или **XML** и задающая значения полей корневого объекта;
- **<Path>** – путь к **lookup-значению**.

В случае использования в **FDS-выражениях** условных операторов, последние транслируются в условные операторы языка **FBL**.

5.2.5 DocumentValidator

В данном контексте **FDS-выражение** транслируется в язык **FBL**. При работе механизма **DocumentValidator** в выражении производится проверка поля на соответствие какому-либо значению и, в случае несоответствия, возникает исключение. Выражение берется из атрибута **EXPRESSION** и транслируется в **FBL** следующим образом:

```
function TDocumentValidator.PrepareScript(ScriptExpression: IXMLDOMElement): String;
var lExpression, lMasterPrefix: String;
begin
 Result := 'uses NXML;' + #10#13;
 Result := Result + 'var Result;' + #10#13;
 Result := Result + 'begin' + #10#13;
 // препарить выражение - убрать { u }
 lMasterPrefix := GetMasterPrefix(ScriptExpression);
 lExpression := ScriptExpression.getAttribute('EXPRESSION');
 if (lMasterPrefix <> '') then
   lExpression := AnsiReplaceStr(lExpression, '{' + lMasterPrefix + '.', '')
 else
```

```
  lExpression := AnsiReplaceStr(lExpression, '{', '');
lExpression := AnsiReplaceStr(lExpression, '}', '');
Result := Result + '  Result := ' + lExpression + ';' + #10#13;
Result := Result + 'end' + #0;
end;
```

FBL-выражение возвращает **true** или **false**, в случае **false** (значение не удовлетворяет выражению) возникает исключение.

6

Аббревиатуры месяцев и дней недели

Таблица 16 – Значения для описателей дней недели

№ п/п	ddd	dddd
1	Пн	Понедельник
2	Вт	Вторник
3	Ср	Среда
4	Чт	Четверг
5	Пт	Пятница
6	Сб	Суббота
7	Вс	Воскресенье

Таблица 17 – Значения для описателей месяцев года

№ п/п	mmm	mmmm
1	янв	января
2	фев	февраля
3	мар	марта
4	апр	апреля
5	май	мая
6	июн	июня
7	июл	июля
8	авг	августа
9	сен	сентября
10	окт	октября
11	ноя	ноября
12	дек	декабря

7

Виды округления

Таблица 18 – Виды округления

Вид округления	Описание	0,35	-0,25	-0,41	-0,59
Математическое	Если отбрасываемый знак меньше 5, то оставляемый знак не меняется. Если отбрасываемый знак больше, либо равен 5, то оставляемый знак увеличивается на единицу.	0,4	-0,3	-0,4	-0,6
Банковское	Аналогично математическому во всех случаях, кроме ситуации, когда отбрасываемый знак равен 5. В этих случаях округление производится к ближайшему четному.	0,4	-0,2	-0,4	-0,6
Округление к нулю	Простое отсечение отбрасываемых знаков.	0,3	-0,2	-0,4	-0,5
Округление от нуля	Симметричное округление в направлении от нуля.	0,4	-0,3	-0,5	-0,6
Округление к плюс-бесконечности	Если у числа округляемые знаки не равны 0, то число округляется в большую сторону.	0,4	-0,2	-0,4	-0,5
Округление к минус-бесконечности	Если у числа округляемые знаки не равны 0, то число округляется в меньшую сторону (в случае положительных чисел округляемые знаки отбрасываются, в случае отрицательных чисел значение числа увеличивается по модулю).	0,3	-0,3	-0,5	-0,6

НАШИ КОНТАКТЫ

Звоните:
(495) 784-70-00

Пишите:
bft@bftcom.com

Будьте с нами online:
www.bftcom.com

Приезжайте:
127018, Москва, ул. Складочная, д.3, стр.1

Дружите с нами в социальных сетях:

vk.com/bftcom

facebook.com/companybft

twitter.com/bftcom

instagram.com/bftcom