ГОСТ Р ИСО 10303-22—2002

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 22

Методы реализации
Стандартный интерфейс доступа к данным

Издание официальное

БЗ 6—2001/134

ГОССТАНДАРТ РОССИИ
Москва

Предисловие

1 РАЗРАБОТАН Всероссийским научно-исследовательским институтом стандартизации (ВНИИстандарт) при участии Научно-технического центра «ИНТЕГРО-Д»

ВНЕСЕН Техническим комитетом по стандартизации ТК 431 «CALS-технологии»

2 ПРИНЯТ И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 25 июня 2002 г. № 250-ст

3 Настоящий стандарт представляет собой аутентичный текст международного стандарта ИСО 10303-22—98 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 22. Методы реализации. Стандартный интерфейс доступа к данным»

4 ВВЕДЕН ВПЕРВЫЕ

© ИПК Издательство стандартов, 2002

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Госстандарта России

Содержание

1 Область применения 1
2 Нормативные ссылки 2
3 Определения и сокращения 2
3.1 Термины, определенные в ГОСТ Р 10303-1 2
3.2 Термины, определенные в ГОСТ Р ИСО 10303-11 2
3.3 Другие определения 2
3.4 Сокращения 3
4 Краткий обзор СИДД (SDAI) 3
4.1 Интерфейсы доступа к данным 3
4.2 Команды (операции) и состояние сеанса 3
4.3 Хранилища, экземпляры схем и СИДД-модели 4
4.4 Сообщения (транзакции) и режимы доступа 5
4.5 Сеанс, словарь данных и управление совокупностью 5
4.6 Схема параметризированных данных СИДД 6
4.7 Функциональные требования (спецификация) 6
4.8 Языковые привязки СИДД 6
4.9 Обработка ошибок 7
5 Основные принципы 7
6 Схема словаря СИДД 8
6.1 Введение 8
6.2 Общие положения и допущения 8
6.3 Определения типов схемы словаря СИДД 8
6.4 Определения объектов схемы словаря СИДД 10
7 Схема сеанса СИДД 21
7.1 Введение 21
7.2 Фундаментальные принципы и допущения 22
7.3 Определения типов схемы сеанса СИДД 22
7.4 Определения объектов схемы сеанса СИДД 22
8 Схема совокупности СИДД 26
8.1 Введение 26
8.2 Фундаментальные принципы и допущения 26
8.3 Определения типов схемы совокупности СИДД 27
8.4 Определения объектов схемы совокупности СИДД 27
9 Схема параметризированных данных СИДД 30
9.1 Введение 30
9.2 Фундаментальные принципы и допущения 31
9.3 Определения типов схемы параметризированных данных СИДД 31
9.4 Определения объектов схемы параметризированных данных СИДД 33
10 Команды СИДД 39
10.1 Введение 39
10.2 Фундаментальные принципы и допущения 40
10.3 Команды среды 41
10.4 Команды сеанса 41
10.5 Команды хранилища 48
10.6 Команды экземпляра схемы 50
10.7 Команды СИДД-модели 55
10.8 Команды области действия 59
10.9 Команды типа 63
10.10 Команды экземпляров объектов 65
10.11 Команды прикладного экземпляра 71
10.12 Команды агрегата экземпляров объекта 80
10.13 Команды агрегата прикладных экземпляров 84
10.14 Команды неупорядоченного набора (коллекции) прикладных экземпляров . . . 85
10.15 Команды упорядоченного набора (коллекции) экземпляров объектов 87

10.16 Команды упорядоченного набора (коллекции) прикладных экземпляров 88
10.17 Команды массива экземпляров объекта 90
10.18 Команды массива прикладных экземпляров 91
10.19 Команды списка прикладных экземпляров 93
11 Ошибки СИДД 97
12 Модель состояния СИДД 99
12.1 Модель состояния для транзакции уровня 1 102
12.2 Модель состояния для транзакции уровня 2 103
12.3 Модель состояния для транзакции уровня 3 104
13 Классы реализации 106
13.1 Реализации СИДД 106
13.2 Спецификация классов реализаций 107
13.3 Команды, необходимые для классов реализаций 109
Приложение А Отображение конструкций языка EXPRESS в конструкции схемы словаря СИДД 111
А.1 Конструкции языка EXPRESS 111
А.2 Информация об эквивалентности области значений 113
Приложение В Форма заявки о соответствии реализации протоколу 115
Приложение С Регистрация информационного объекта 116
С.1 Обозначение документа 116
С.2 Обозначение схемы 116
Приложение D Диаграммы на языке EXPRESS-G 116
Приложение Е Распечатки (листинги) схем СИДД на языке EXPRESS 126
Предметный указатель 127

Введение

Стандарты серии ГОСТ Р ИСО 10303 распространяются на машинноориентированное представление данных об изделии и обмен этими данными. Целью является создание механизма, позволяющего описывать данные об изделии на протяжении всего жизненного цикла изделия независимо от конкретной системы. Характер такого описания делает его пригодным не только для обмена инвариантными файлами, но также и для создания баз данных об изделиях, коллективного пользования этими базами и архивации соответствующих данных.

Серия ГОСТ Р ИСО 10303 представляют собой набор отдельно издаваемых стандартов (частей). Части данной серии стандартов относятся к одной из следующих тематических групп: методы описания, интегрированные ресурсы, прикладные протоколы, комплекты абстрактных тестов, методы реализации и аттестационное тестирование.

Настоящий стандарт входит в серию ГОСТ Р ИСО 10303 и определяет стандартный интерфейс доступа к данным (СИДД — SDAI), описанный средствами языка EXPRESS (ГОСТ Р ИСО 10303-11). Команды, определенные в стандарте, позволяют прикладному программисту манипулировать данными с использованием СИДД, опираясь на их описание в конкретной(ных) схеме(ах). Стандартизация интерфейса доступа к данным и описаний данных обеспечивает функциональные возможности для интеграции различных компонентов программных средств, получаемых от разных поставщиков.

Основными тематическими частями настоящего стандарта являются:

- конструктивы среды СИДД, определяемые на языке EXPRESS (разделы 6—9);
- команды, ошибки и состояния СИДД (разделы 10—12);
- классы реализации функциональных возможностей СИДД, которым должны соответствовать реализации (раздел 13).

Прикладные вычислительные системы реализуются с использованием машинных языков. При установлении требований к функциональным возможностям, определенным в настоящем стандарте, в конкретном машинном языке используют понятие языковой привязкой СИДД (SDAI language binding). Так как существует много машинных языков, возможно существование множества языковых привязок СИДД. Языковые привязки СИДД для конкретных машинных языков определяются в других стандартах серии ГОСТ Р ИСО 10303.

Реализации конкретных языковых привязок СИДД не должны обеспечивать полного набора функциональных возможностей, описанных в настоящем стандарте. Конкретные наборы функциональных возможностей сгруппированы в классы реализации. Классы реализации, по которым определяется соответствие настоящему стандарту, установлены в разделе 13.

Примечания

1 Настоящий стандарт дополнен приложениями A, B, C, D и E.

2 В настоящем стандарте конструктивы, описанные с использованием языка EXPRESS, в ряде случаев выделены полужирным шрифтом (например **referenced_item**).

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 22

Методы реализации. Стандартный интерфейс доступа к данным

Industrial automation systems and integration. Product data representation and exchange.
Part 22. Implementation methods. Standard data access interface

Дата введения 2003—07—01

1 Область применения

Настоящий стандарт определяет функциональные характеристики интерфейса доступа к данным. На интерфейс ссылаются как на стандартный интерфейс доступа к данным (СИДД—SDAI). СИДД определяет команды, доступные приложениям в целях получения и управления данными, структура которых определена с использованием языка EXPRESS (ГОСТ Р ИСО 10303-11).

СИДД описан в терминах, независящих от любого машинного языка или системы. При установлении требований к функциональным возможностям, определенным в настоящем стандарте, в конкретном машинном языке используют понятие языковой привязкой СИДД (SDAI language binding). Языковые привязки СИДД для конкретных машинных языков определяются в группе стандартов серии ГОСТ Р ИСО 10303 по методам реализации.

Настоящий стандарт распространяется на:

- доступ и манипулирование экземплярами объектов, описанными с использованием языка определения данных EXPRESS;
- одновременный доступ отдельного приложения к многим хранилищам данных;
- возможности для объединения операций в группы, воздействие которых может быть сохранено или отменено по усмотрению приложения;
- доступ к словарю, описывающему элементы данных, которыми может манипулировать приложение;
- способность вызывать проверку правильности ограничений, установленных с использованием языка EXPRESS, по усмотрению приложения;
- обеспечение управления отношениями зависимости между экземплярами объектов;
- возможности описания логических коллекций экземпляров объектов, определяющих совокупность, в которой допускаются ссылки между экземплярами объектов;
- возможности описания логических коллекций экземпляров объектов, определяющих совокупность, в которой действуют глобальные правила;
- обеспечение использования данных, созданных в контексте одной схемы, в контексте другой схемы.

Настоящий стандарт не распространяется на:

- полную спецификацию поведения реализаций СИДД в многопользовательской среде.

Примечание 1 — Это не препятствует реализаций СИДД обеспечивать многопользовательский разделяемый доступ к данным, когда поведение реализации зависит от определенной технологии хранения данных;

- конкретное обеспечение установления соединения с удаленным хранилищем данных.

Примечание 2 — Это не препятствует реализаций СИДД обеспечивать доступ к удаленному хранилищу данных через механизм, специфический для данной реализации;

Издание официальное

- команды доступа к данным и манипулирования ими, зависящие от семантики данных;
- требования к механизмам или форматам, посредством которых данные представляются в хранилище;
- создание, удаление и обозначение хранилищ данных, доступных через СИДД.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие стандарты.

ГОСТ Р ИСО/МЭК 8824-1—2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации

ГОСТ Р ИСО 10303-1—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы.

ГОСТ Р ИСО 10303-11—2000 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS

ГОСТ Р ИСО 10303-21—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 21. Методы реализации. Кодирование открытым текстом структуры обмена

ГОСТ Р ИСО 10303-31—2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 31. Методология и основы аттестационного тестирования. Общие положения

ИСО 8601—88* Элементы данных и форматы обмена. Информационный обмен. Представление дат и времени

3 Определения и сокращения

3.1 Термины, определенные в ГОСТ Р 10303-1

В настоящем стандарте использованы следующие термины:

- **приложение;**
- **прикладной протокол (ПП);**
- **данные;**
- **метод реализации;**
- **информация;**
- **информационная модель;**
- **информационная модель изделия;**
- **заявка о соответствии реализации протоколу (ЗСРП);**
- **структура.**

3.2 Термины, определенные в ГОСТ Р ИСО 10303-11

В настоящем стандарте использованы следующие термины:

- **тип данных сложного объекта;**
- **тип данных;**
- **объект;**
- **тип данных объекта;**
- **экземпляр объекта;**
- **экземпляр;**
- **совокупность;**
- **значение.**

3.3 Другие определения

В настоящем стандарте использованы следующие термины с соответствующими определениями.

3.3.1 **прикладная схема** (application schema): Информационная модель, определенная при помощи языка EXPRESS, описывающая данные, представляющие интерес в конкретном контексте или области.

---

* Оригиналы стандартов ИСО — во ВНИИКИ Госстандарта России.

Примечание — Схемы, определенные в стандартах серии ГОСТ Р ИСО 10303 по прикладным протоколам, должны рассматриваться как прикладные схемы.

3.3.2 **параллельный доступ** (concurrent access): Возможность для нескольких компьютерных приложений одновременно манипулировать данными в хранилище.

3.3.3 **ограничение** (constraint): Описанный средствами языка EXPRESS предел, налагаемый на данные, позволяющий оценить допустимость данных для использования в конкретном контексте.

3.3.4 **текущая схема** (current schema): EXPRESS-схема, внутри которой элементы из других схем становятся видимыми посредством EXPRESS-спецификации интерфейса.

3.3.5 **внешняя схема** (external schema): EXPRESS-схема со всеми разрешенными импортируемыми элементами, содержащая типы данных, определенные как имеющие область значений, эквивалентную типам данных из собственной схемы.

Примечание — В А.1.1 описан процесс разрешения импортирования элементов.

3.3.6 **инородная схема** (foreign schema): EXPRESS-схема, отличная от текущей, элементы которой становятся видимыми в текущей схеме посредством EXPRESS-спецификации интерфейса.

3.3.7 **идентификатор** (identifier): Зависимые от реализации обозначения экземпляра объекта и агрегата, обеспечивающие их уникальность в течение сеанса СИДД.

3.3.8 **класс реализации** (implementation class): Спецификация реализуемого подмножества функциональных возможностей, определенных в настоящем стандарте, которому может соответствовать реализация.

3.3.9 **итератор** (iterator): Механизм, позволяющий прикладной программе «пробежать» по содержанию экземпляра агрегата.

3.3.10 **собственная схема** (native schema): EXPRESS-схема со всеми разрешенными импортируемыми элементами, на которой могут базироваться экземпляр схемы или СИДД-модели.

3.3.11 **хранилище** (repository): Определенное средство хранения данных.

3.3.12 **экземпляр схемы** (schema instance): Логическое объединение связанных моделей СИДД, устанавливающее область значений экземпляров объектов. Эта область ограничивает ссылки между экземплярами объектов и является областью, в которой проверяют глобальные правила.

3.3.13 **СИДД-модель** (SDAI-model): Контейнер, внутри которого существуют связанные экземпляры объектов.

3.3.14 **языковая привязка СИДД** (SDAI language binding): Функциональные возможности СИДД, определенные на конкретном машинном языке.

3.3.15 **схема СИДД** (SDAI schema): EXPRESS-схема, определенная в настоящем стандарте.

3.3.16 **сеанс** (session): Команды (операции), которые выполняются между началом и окончанием использования реализации СИДД одним приложением.

3.3.17 **проверка правильности** (validation): Проверка экземпляров на соответствие ограничениям, установленным в схеме, описывающей их структуру, значения и отношения.

3.4 Сокращения

В настоящем стандарте использованы следующие сокращения:

ПП (AP) — прикладной протокол;
ЗСРП (PICS) — заявка о соответствии реализации протоколу;
ТЧ (RO) — только чтение;
ЧЗ (RW) — чтение—запись;
СИДД (SDAI) — стандартный интерфейс доступа к данным.

4 Краткий обзор СИДД (SDAI)

4.1 Интерфейсы доступа к данным

Язык EXPRESS позволяет определять объекты с атрибутами и ограничения, которым должна удовлетворять допустимая совокупность этих объектов. СИДД устанавливает требования к программному интерфейсу для создания и манипулирования экземплярами EXPRESS-объектов. СИДД и язык EXPRESS совместно определяют интерфейс доступа к данным, который не зависит от конкретной технологии хранения данных.

4.2 Команды (операции) и состояние сеанса

С началом сеанса СИДД его команды могут быть использованы для манипуляции экземплярами типов данных объектов, определенными в приложении и схемах СИДД. Сеанс имеет несколько

различных состояний, представленных в табличной форме в разделе 12. В каждом состоянии доступен ряд команд (операций) СИДД, некоторые из которых могут изменять данное состояние. Информация, связанная с сеансом и его состоянием, доступна в течение сеанса в виде сочетания схем сеанса и совокупности СИДД (см. разделы 7 и 8).

4.3 Хранилища, экземпляры схем и СИДД-модели

СИДД определяет интерфейс между приложением и средой, в которой существуют экземпляры объектов. Два аспекта этой среды известны как хранилища и экземпляры схемы. Хранилищами являются средства хранения данных. Экземплярами схем являются логические коллекции СИДД-моделей, из которых может быть получено множество экземпляров объектов. Это множество экземпляров объектов является областью, в которой обеспечиваются ссылки между экземплярами объектов и проверка глобальных правил. Несмотря на то, что экземпляры схем, подобных СИДД-моделям, создаются внутри хранилища, СИДД-модели из любого другого хранилища могут быть связаны с данным экземпляром схемы.

Примечание — Хранилище может быть реализовано в памяти как одиночная база данных, многократные базы данных, одиночный файл, набор файлов или в любой другой форме.

Пример 1 — На рисунке 1 показаны некоторые взаимосвязи между СИДД-моделями, хранилищами и экземплярами схем. Экземпляры схем 1А и 1В и СИДД-модели 11—13 базируются на схеме 1, однако они существуют в различных хранилищах. Ссылки между СИДД-моделями 11 и 13 не разрешены, так как эти модели не связаны с одним и тем же экземпляром схемы. СИДД-модель 13, базирующаяся на схеме 1, связана с экземпляром схемы 2А, базирующимся на схеме 2. Чтобы это было возможно, по крайней мере одна пара типов объектов должна быть объявлена как область, эквивалентная двум схемам в словаре данных СИДД.

Экземпляры объектов создаются в СИДД-моделях, образуемых в хранилищах. Экземпляры объектов, составляющие каждую СИДД-модель, базируются на единственной EXPRESS-схеме с разрешенными интерфейсными спецификациями. Экземпляры объектов в одной СИДД-модели могут ссылаться на экземпляры объектов в другой СИДД-модели, если обеспечено существование экземпляра схемы, с которой связаны обе эти СИДД-модели. Две СИДД-модели должны основываться на одной и той же EXPRESS-схеме или двух EXPRESS-схемах, которые определены как имеющие конструкции эквивалентности областей (см. А.2). СИДД-модель может быть связана с несколькими экземплярами схемы.

Рисунок 1 — Пример структуры хранения СИДД

4.4 Сообщения (транзакции) и режимы доступа

Уровни реализации СИДД определяют обеспечиваемые ими сообщения. Сообщение состоит из рядов команд (операций), воздействие которых может быть сохранено или отменено как единое целое. Также доступны средства, обеспечивающие приложению возможность управления доступом к конкретным хранилищам и СИДД-моделям. Сообщения и доступ к СИДД-моделям содержат связанные с ними режимы: «чтение—запись» и «только чтение». Режим «чтение—запись» допускает команды доступа, создания, обновления и удаления экземпляров в СИДД-моделях и хранилищах. Режим «только чтение» не допускает команды создания, обновления или удаления экземпляров в СИДД-моделях и хранилищах. СИДД-модель не может быть доступна в режиме «чтение—запись», когда инициировано сообщение в режиме «только чтение».

4.5 Сеанс, словарь данных и управление совокупностью

Схема сеанса СИДД (см. раздел 7) описывает структуру сеанса СИДД. Схема совокупности СИДД (см. раздел 8) описывает упорядоченные структуры, доступные для управления совокупностью, основанной на схеме. Схема совокупности СИДД определяет организационные объекты, которые приложение может создавать в течение сеанса.

Для обеспечения приложениям необходимого доступа к информации о схеме, определяющей прикладные данные, СИДД создает словарь данных. Схема словаря СИДД (см. раздел 6) описывает структуру словаря данных. Словарь данных состоит из набора экземпляров объектов, определенных в схеме словаря СИДД. Так как не все приложения требуют доступа к словарю данных, то класс реализации может быть определен без обеспечения требуемого словаря данных (см. 13.1.2). В этом случае прикладному программисту необходимы полное знание схемы и возможность ссылаться на элементы схемы по их именам.

Примечание — На рисунке 2 в упрощенных терминах показана взаимосвязь между прикладными данными, словарем, сеансом и совокупностью организационных данных, а также между прикладной программой и реализацией СИДД. На рисунке 2 также показано, для каких типов данных прикладная программа имеют доступ в режимах «чтение—запись» или «только чтение».

Рисунок 2 — Взаимосвязи элементов архитектуры данных СИДД

Создание, удаление и изменение данных сеанса, организация совокупности и словаря проявляются как последовательность специальных команд, предназначенных для управления средой СИДД. Команды экземпляра объекта используются для доступа к данным сеанса, организации совокупности и словаря так же, как и для экземпляров приложения. Не все команды экземпляра объекта необходимы для обеспечения доступа к данным организации совокупности, поскольку экземпляры схемы совокупности СИДД не обязательно присутствуют в рамках СИДД-модели (см. 8.1).

4.6 Схема параметризированных данных СИДД

Схема параметризированных данных СИДД (см. раздел 9) в абстрактных терминах описывает различные типы данных экземпляра, передаваемые через интерфейс. Это обеспечивает возможность определения команд СИДД. Реализация СИДД не обязательно включает полную схему параметризированных данных. Эта схема определяет отношения подтипов между типами экземпляров объекта, которые должна обеспечивать реализация СИДД. Другие характеристики этих экземпляров не определяются в данной схеме, так как они зависят от реализации.

4.7 Функциональные требования (спецификация)

СИДД определяет функциональные требования к набору команд (операций) для запроса данных и манипулирования ими. Команды СИДД подразделяют на несколько категорий:

- команды среды, устанавливающие сеанс СИДД (см. 10.3);
- команды сеанса, позволяющие приложению управлять сообщениями (транзакциями), хранилищами и запросами в сеансе (см. 10.4);
- команды хранилища, позволяющие приложению управлять доступом к СИДД-моделям внутри хранилищ (см. 10.5);
- команды экземпляров схем, позволяющие приложению управлять связями СИДД-моделей с экземплярами схем, проверять правильность глобальных EXPRESS-правил и осуществлять ссылки внутри экземпляра схемы (см. 10.6);
- команды СИДД-моделей, позволяющие приложению создавать экземпляры и управлять доступом к СИДД-модели (см. 10.7);
- команды, позволяющие приложению создавать зависимые отношения между экземплярами объектов и управлять ими (см. 10.8).

Примечание — Отношения зависимости базируются на конструкции SCOPE (область применения), описанной в ГОСТ Р ИСО 10303—21;

- команды, позволяющие приложению проверять информацию о типе данных и эквивалентности областей значений (см. 10.9);
- команды экземпляра объекта, позволяющие приложению манипулировать экземплярами типов данных объектов, создаваемых в схемах, определяющих СИДД, и прикладных схемах (см. 10.10);
- команды прикладного экземпляра, позволяющие приложению создавать, изменять, удалять и проверять допустимость экземпляров типов данных объектов и создавать экземпляры агрегатов, определенных в прикладных схемах (см. 10.11);
- категории команд прикладного экземпляра и экземпляра объекта, позволяющие приложению создавать, изменять, удалять экземпляры различных типов агрегатов, а также манипулировать ими (см. 10.12—10.19).

Команды классифицированы таким образом, что все команды, обрабатывающие определенные виды объектов, описаны в подразделе, связанном с данным объектом. Исключением из этой классификации являются команды, создающие, открывающие или инициирующие различные СИДД и прикладные объекты. Поскольку эти команды обычно влияют на свойства объектов более высокого уровня, они рассматриваются как команды объекта, внутри которого создается, открывается или инициируется текущий объект.

Пример 2 — Открытие хранилища является командой сеанса, а не командой хранилища, так как она изменяет свойства сеанса.

В спецификации каждой команды СИДД перечисляются требуемые входные и выходные параметры. Типы этих параметров выбираются из схем словаря, сеанса, совокупности и параметризованных данных СИДД.

4.8 Языковые привязки СИДД

В настоящем стандарте команды СИДД определены независимо от какого-либо языка программирования. Для этих команд разработаны языковые привязки СИДД к машинным языкам (языкам программирования), определяющие функциональные возможности, обеспечиваемые реализациями. Языковые привязки СИДД описаны в других стандартах серии ГОСТ Р ИСО 10303 — в группе методов реализации. Конкретная реализация СИДД должна обеспечивать команды, описанные в соответствующих языковых привязках СИДД.

Языковые привязки СИДД обеспечивают команды, описанные в разделе 10, но может отсутствовать однозначное соответствие между этими командами и функциями или подпрограммами, определенными в языковой привязке СИДД. Языковая привязка СИДД может расширять или нара-

щивать функциональные возможности, определенные в разделе 10, обеспечивая более эффективную и удобную реализацию этих команд.

Различают два типа языковых привязок СИДД: позднюю и раннюю привязку.

Поздние привязки применяют для любой прикладной схемы одним и тем же способом, используя один и тот же набор функций. Этот набор доступен прикладному программисту независимо от прикладной схемы.

Обращения к конкретным конструктивам на языке EXPRESS обеспечиваются через входные параметры, передаваемые командами СИДД. Поздние привязки определяет набор команд СИДД, явно доступных прикладному программисту.

Примечание 1 — Эти команды и их параметры обычно вычисляют во время выполнения требуемой функции.

Пример 3 — Команда набора значений атрибута «**a1**» экземпляра объекта «**i1**» типа «**t1**» в поздней привязке может быть записана как **SetValue(i1, t1.a1, value).**

Ранняя привязка описывает создание интерфейса доступа к данным, основанного на конкретной прикладной схеме. Вместо обеспечения СИДД через входные параметры команд, обращения к конкретным конструктивам на языке EXPRESS могут явно или неявно встраиваться в имена функций или подпрограмм. Полный набор функций, доступных прикладному программисту, зависит от прикладной схемы, лежащей в основе реализации СИДД.

Примечание 2 — Возможна компиляция прикладной схемы для создания реализации. Такой компилятор обычно генерирует функции и параметры, вычисляя их во время компиляции прикладной программы.

Пример 4 — Команда набора значений атрибута «**a1**» экземпляра объекта «**i1**» типа «**t1**» в ранней привязке должна быть записана как **SetValuet1a1(i1, value).**

Ранние и поздние языковые привязки СИДД могут быть опубликованы вместе в одном стандарте серии ГОСТ Р ИСО 10303 — в группе методов реализации. Реализации СИДД могут обеспечивать одновременное выполнение ранней и поздней привязок и их использование в одной прикладной программе.

4.9 Обработка ошибок

Описание каждой команды включает список возможных индикаторов ошибок, могущих возникать при условиях, препятствующих успешному завершению данной команды. В настоящем стандарте не определено, как выдаются уведомления об ошибках для отдельных команд СИДД. Каждая языковая привязка СИДД определяет механизм выдачи уведомления об ошибке, соответствующий данному конкретному языку.

Схема сеанса СИДД определяет содержание уведомления об ошибке через атрибут **sdai_session.errors,** который доступен в качестве ресурса любому механизму обработки ошибок языковой привязки СИДД и прикладному программисту.

Полный список индикаторов ошибок языковых привязок СИДД приведен в разделе 11 для соответствующих конструктивов конкретного языка. Реализация должна выбирать коды ошибок из раздела 11 в качестве значения атрибута **error_event.error** (см. 7.4.7), если команда СИДД ошибочна, за исключением команды открытия сеанса (см. 10.3.1), которая не может присваивать значение атрибуту, так как сеанс еще не инициирован. Механизм уведомления об ошибке, установленный в каждой языковой привязке СИДД, не обязательно выдает коды ошибок, идентичные по типу или значению кодам ошибок, выбранным в атрибуте **error_event.error**. Привязка может определить дополнительные функции или подпрограммы для уведомления об ошибках и управления ими. Любая такая функция или подпрограмма не может изменять состояний ошибок сеанса СИДД.

5 Основные принципы

В настоящем стандарте использованы следующие предпосылки и допущения:

- во время сеанса СИДД для каждого экземпляра объекта или агрегата, описанного в схеме СИДД или прикладной схеме, доступен уникальный и неизменный идентификатор;
- EXPRESS-схема, на которой основаны СИДД-модели и экземпляры схем, является полностью развернутой формой схемы, в которой допускается использование всех элементов, импортируемых из других схем, что делает ее полной без ссылок на другие схемы (см. А.1);

- процесс, который делает совокупность схемы словаря СИДД доступной в сеансе, разворачивает любые типы данных сложных объектов в виде результата или явного или неявного применения ограничения ANDOR или AND, в соответствии с А.1.3;

- в соответствии с требуемым классом реализации, реализации СИДД делают доступной совокупность схемы словаря СИДД, с помощью элементов на языке EXPRESS, определенных в схеме сеанса СИДД и прикладных схемах;

- схемы, определенные в разделах 6—9, не являются прикладными протоколами или информационными моделями изделий, а учитывают эффективность и удобство описания команд СИДД;

- экземпляры объектов, основанные на одной схеме, могут быть распространены в контексты других схем, базируясь на эквивалентности областей значения (см. А.2) и конструктивах эквивалентности областей значения, определенных в схеме словаря СИДД (см. 6.4.1, 6.4.8 и 6.4.9);

- проверки правильности EXPRESS-ограничений проводят только по запросу приложения. При проведении данных проверок принимают во внимание все создания, изменения и удаления СИДД-моделей, экземпляров схем, конструкций SCOPE (области применения) и прикладных экземпляров.

6 Схема словаря СИДД

Следующее объявление на языке EXPRESS открывает схему словаря СИДД.

EXPRESS-спецификация

```
*)
SCHEMA SDAI_dictionary_schema;
(*
```

6.1 Введение

Схема словаря СИДД определяет структуру словаря данных, позволяющего получить информацию об EXPRESS-схемах, описывающих экземпляры, используемые в сеансе СИДД. Структура схемы словаря СИДД отражает структуру самого языка EXPRESS. Но в схеме словаря СИДД отражены не все элементы, определяемые с использованием языка EXPRESS, так как некоторые элементы не требуются для определения команд СИДД. Отображение EXPRESS-схем в совокупность схемы словаря СИДД описано в приложении А.

Совокупность схемы словаря СИДД, основанная на схеме, известной для реализации СИДД как часть словаря данных, должна существовать в собственной отдельной СИДД-модели. Прикладные схемы и схема сеанса СИДД должны присутствовать как часть словаря данных.

Должен существовать экземпляр схемы, основанный на схеме сеанса СИДД, вместе с которым СИДД-модели составляют словарь данных для прикладных схем и схемы сеанса СИДД. СИДД-модели, составляющие словарь данных СИДД, должны быть доступны в режиме «только чтение». От реализации СИДД не требуется сохранения данных словаря в описанном виде. Однако команды СИДД должны обеспечивать доступ к данным словаря так, как будто данные структурированы в описанном выше виде.

6.2 Общие положения и допущения

Структура объектов и атрибутов схемы словаря СИДД учитывает реализацию и эффективность доступа.

Пример 5 — Набор (коллекция) **schema_definition.entities** включает все типы данных объектов, определенных локально в текущей схеме, а также типы, импортированные в данную схему.

Схема словаря СИДД обеспечивает неизменность информации, содержащейся в спецификации интерфейса языка EXPRESS (см. А.1.1).

Поскольку схема словаря СИДД разработана для наполнения допустимыми, в соответствии с ГОСТ Р ИСО 10303-11, EXPRESS-схемами, все ограничения, управляющие совокупностью данной схемы, в ней не определяются. Для совокупности схемы словаря СИДД должны быть обеспечены ограничения, установленные в ГОСТ Р ИСО 10303-11. В частности, в схеме словаря СИДД не определены ограничения на EXPRESS-идентификаторы для элементов, объявленных в схеме, включая самою схему.

6.3 Определения типов схемы словаря СИДД

Данный подраздел устанавливает общие положения для словаря, определяющие применение в нем конструктива TYPE из языка EXPRESS.

6.3.1 Тип **base_type**

Тип **base_type** представляет собой выбор из типов **simple_type, aggregation_type** или **named_type.** Данный тип определяет типы данных, которые могут быть использованы в качестве значений для атрибута или как элемент агрегата.

EXPRESS-спецификация

```
*)
TYPE base_type = SELECT
    (simple_type,
    aggregation_type,
    named_type);
END_TYPE;
(*
```

6.3.2 Тип **constructed_type**

Тип **constructed_type** представляет собой выбор из типов **enumeration_type** или **select_type**. Данный тип определяет типы данных с синтаксической структурой, используемые для обеспечения представления определяемых типов данных из языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE constructed_type = SELECT
    (enumeration_type,
    select_type);
END_TYPE;
(*
```

6.3.3 Тип **underlying_type**

Тип **underlying_type** представляет собой выбор из типов **simple_type, aggregation_type**, **defined_type** или **constructed_type**. Данный тип определяет типы данных, используемые для обеспечения представления определяемых типов данных из языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE underlying_type = SELECT
    (simple_type,
    aggregation_type,
    defined_type,
    constructed_type);
END_TYPE;
(*
```

6.3.4 Тип **type_or_rule**

Тип **type_or_rule** представляет собой выбор из типов **named_type** или **global_rule**. Данный тип описывает определение или ограничение совокупности.

EXPRESS-спецификация

```
*)
TYPE type_or_rule = SELECT
    (named_type,
    global_rule);
END_TYPE;
(*
```

6.3.5 Тип **explicit_or_derived**

Тип **explicit_or_derived** представляет собой выбор из типов **explicit_attribute** или **derived_attribute**. Данный тип определяет атрибуты, которые могут быть переопределены как вычисляемые (см. 9.2.3.4 ГОСТ Р ИСО 10303-11).

EXPRESS-спецификация

```
*)
TYPE explicit_or_derived = SELECT
    (explicit_attribute,
    derived_attribute);
END_TYPE;
(*
```

6.3.6 Тип **express_id**

Тип **express_id** является EXPRESS-идентификатором (см. 7.4 ГОСТ Р ИСО 10303-11) для элементов, объявленных в EXPRESS-схеме. Хотя ГОСТ Р ИСО 10303-11 устанавливает, что регистр букв не имеет значения для EXPRESS-идентификаторов, в качестве значений атрибутов схемы словаря СИДД, область значений которых определяется типом **express_id**, должны использоваться строчные буквы.

EXPRESS-спецификация

```
*)
TYPE express_id = STRING;
END_TYPE;
(*
```

6.3.7 Тип **info_object_id**

Тип **info_object_id** является однозначным идентификатором информационного объекта для EXPRESS-схемы в открытой системе (см. приложение С).

EXPRESS-спецификация

```
*)
TYPE info_object_id = STRING;
END_TYPE;
(*
```

6.4 Определения объектов схемы словаря СИДД

Данный подраздел устанавливает общие положения для словаря, определяющие применение в нем конструктива ENTITY из языка EXPRESS.

6.4.1 Объект **schema-definition**

Объект **schema-definition** является представлением SCHEMA из языка EXPRESS и конструктивом, на которой основаны СИДД-модели и экземпляры схем, и определяет область применения (действия) для набора (коллекции) описаний объектов, типов и правил, состоящих из описаний, взятых из текущей EXPRESS-схемы и разрешенных в этой схеме по спецификации интерфейса на языке EXPRESS. Элементы из инородных схем допускаются в текущей схеме в соответствии с А.1.1.

EXPRESS-спецификация

```
*)
ENTITY schema_definition;
   name              : express_id;
   identification    : OPTIONAL info_object_id;
INVERSE
      entities              : SET [0:?] OF entity_definition FOR parent_schema;
      types                 : SET [0:?] OF defined_type FOR parent_schema;
      global_rules          : SET [0:?] OF global_rule FOR parent_schema;
      external_schemas      : SET [0:?] OF external_schema FOR native_schema;
UNIQUE
      UR1 : identification;
END_ENTITY;
(*
```

Определения атрибутов

name — имя схемы;

identification— идентификатор информационного объекта схемы, основанный на **schema_definition** (при его наличии);

entities — объекты, объявленные или разрешенные в схеме;

types — типы, объявленные или разрешенные в схеме;

global_rules — глобальные правила, объявленные или разрешенные в схеме;

external_schemas — схемы, содержащие типы, определенные как имеющие эквивалентные области значений с типами из данной схемы.

Формальные утверждения

UR1 — обозначение объекта для схемы должно быть уникальным.

6.4.2 Объект **interface_specification**

Объект **interface_specification** является представлением в текущей схеме элементов, изначально объявленных в инородной схеме (см. раздел 11 ГОСТ Р ИСО 10303-11). Все элементы, импортиро-

ванные явно (через операторы USE и/или REFERENCE) или неявно из конкретной инородной схемы, должны появиться в том же экземпляре объекта **interface_specification**. Операторы USE или REFERENCE, примененные ко всей схеме полностью, явно импортируют все элементы, объявленные в данной схеме.

EXPRESS-спецификация

```
*)
ENTITY interface_specification;
    current_schema_id     : express_id;
    explicit_items        : SET[1:?] OF explicit_item_id;
    implicit_items        : SET[0:?] OF implicit_item_id;
END_ENTITY;
(*
```

Определения атрибутов

current_schema_id — имя текущей схемы, в которую импортируются элементы; схема, в которой объявлены спецификации USE или REFERENCE;

explicit_items — элементы, импортируемые при помощи операторов USE или REFERENCE языка EXPRESS;

implicit_items — неявно импортируемые элементы.

6.4.3 Объект **interfaced_item**

Объект **interfaced_item** является элементом, определенным в инородной схеме, доступным для текущей схемы через спецификацию интерфейса на языке EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY interfaced_item
    ABSTRACT SUPERTYPE OF (ONEOF (explicit_item_id, implicit_item_id));
    foreign_schema_id : express_id;
END_ENTITY;
(*
```

Определение атрибута

foreign_schema_id — имя схемы, из которой импортирован элемент.

6.4.4 Объект **explicit_item_id**

Объект **explicit_item_id** является объектом **named_type**, определенным в инородной схеме, явно доступным для текущей схемы через операторы USE или REFERENCE языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY explicit_item_id
    ABSTRACT SUPERTYPE OF (ONEOF (used_item, referenced_item))
    SUBTYPE OF (interfaced_item);
    local_definition  : named_type;
    original_id       : OPTIONAL express_id;
END_ENTITY;
(*
```

Определение атрибутов

local_definition — определение в текущей схеме именованного типа, импортированного из инородной схемы.

original_id — именованный тип в инородной схеме (при его наличии). Тип был переименован в спецификации интерфейса.

6.4.5 Объект **used_item**

Объект **used_item** является объектом **explicit_item_id**, доступным текущей схеме через спецификацию интерфейса на языке EXPRESS для оператора USE.

EXPRESS-спецификация

```
*)
ENTITY used_item
    SUBTYPE OF (explicit_item_id);
END_ENTITY;
(*
```

6.4.6 Объект **referenced_item**

Объект **referenced_item** является объектом **explicit_item_id**, доступным текущей схеме через спецификацию интерфейса на языке EXPRESS для оператора REFERENCE.

EXPRESS-спецификация

```
*)
ENTITY referenced_item
    SUBTYPE OF (explicit_item_id);
END_ENTITY;
(*
```

6.4.7 Объект **implicit_item_id**

Объект **implicit_item_id** является объектом **named_type** или **global_rule**, неявно импортированным в текущую схему через спецификацию интерфейса на языке EXPRESS для операторов USE или REFERENCE (см. 11.4 ГОСТ Р ИСО 10303-11).

EXPRESS-спецификация

```
*)
ENTITY implicit_item_id
    SUBTYPE OF (interfaced_item);
    local_definition : type_or_rule;
END_ENTITY;
(*
```

Определение атрибута

local_definition — определение в текущей схеме элемента на языке EXPRESS, неявно импортированного из инородной схемы.

6.4.8 Объект **external_schema**

Объект **external_schema** является EXPRESS-схемой, типы из которой объявляются как имеющие эквивалентные области значений с типами из собственной схемы.

EXPRESS-спецификация

```
*)
ENTITY external_schema;
    name            : express_id;
    native_schema   : schema_definition;
INVERSE
    for_types : SET [1:?] OF domain_equivalent_type FOR owner;
END_ENTITY;
(*
```

Определения атрибутов

name: — имя внешней схемы;

native_schema — локальная схема, в которой типы, определенные во внешней схеме, имеют эквивалентные области значений;

for_types — типы в собственной и внешней схемах, определенные эквивалентными по области значений.

Неформальные утверждения

IP1 — должен существовать отдельный экземпляр внешней схемы для каждой схемы, содержащей типы объектов, объявленные эквивалентными по области значений с типами объектов в собственной схеме.

6.4.9 Объект **domain_equivalent_type**

Объект **domain_equivalent_type** связывает имя объекта **named_type** из внешней схемы с объектом **named_type** из собственной схемы. Этим объявляется, что объект **named_type**, определенный во внешней схеме, эквивалентен по области значения с объектом **named_type**, определенным в собственной схеме.

EXPRESS-спецификация

```
*)
ENTITY domain_equivalent_type;
    external_type_id        : express_id;
    native_type             : named_type;
    owner                   : external_schema;
END_ENTITY;
(*
```

Определения атрибутов

external_type_id — имя типа во внешней схеме, который объявлен эквивалентным по области значений с типом в собственной схеме;

native_type — определение типа в собственной схеме, для которого объявлен тип, эквивалентный по области значений;

owner — внешняя схема, содержащая определение текущего типа, эквивалентного по области значений.

Неформальные утверждения

IP1 — экземпляр объекта **domain_equivalent_type** должен существовать для каждого типа во внешней схеме, эквивалентного по области значений с собственным типом.

IP2 — в случае, когда собственный или внешний тип, или оба одновременно являются типом **defined_type**, основной тип (**underlying_type**) типа **defined_type** должен выражаться типом выбора (**select_type**), содержащим как минимум один тип объекта в качестве элемента выбора.

6.4.10 Объект **named_type**

Объект **named_type** является типом данных языка EXPRESS, которому присвоено имя и который может иметь соответствующие области значения правил.

EXPRESS-спецификация

```
*)
ENTITY named_type
    ABSTRACT SUPERTYPE OF (ONEOF (entity_definition, defined_type));
    name            : express_id;
    where_rules     : LIST [0:?] OF where_rule;
    parent_schema   : schema_definition;
END_ENTITY;
(*
```

Определения атрибутов

name — имя типа данных;

where_rules — области значения правил, определенные в объявлении типа данных в порядке их появления в данном объявлении;

parent_schema — объект **schema_definition,** с которым в словаре данных связан **named_type**.

6.4.11 Объект **defined_type**

Объект **defined_type** является объектом **named_type**, устанавливающим тип вследствие его объявления в операторе TYPE языка EXPRESS, имеет имя и область значения.

EXPRESS-спецификация

```
*)
ENTITY defined_type
    SUBTYPE OF (named_type);
    domain : underlying_type;
END_ENTITY;
(*
```

Определение атрибута

domain — основной тип определяемого типа.

6.4.12 Объект **entity_definition**

Объект **entity_definition** является объектом **named_type**, определяющим объект в соответствии с объявлением ENTITY языка EXPRESS или отображением, применяемым к комбинации объявлений ENTITY языка EXPRESS, ограничивающей использование ключевых слов ANDOR или AND языка EXPRESS (см. А.1.3). Объекты, установленные таким отображением, рассматриваются как подтипы составных типов объектов.

EXPRESS-спецификация

```
*)
ENTITY entity_definition
    SUBTYPE OF (named_type);
    supertypes      : LIST[0:?] OF UNIQUE entity_definition;
    complex         : BOOLEAN;
    instantiable    : BOOLEAN;
    independent     : BOOLEAN;
```

```
INVERSE
    attributes          : SET[0:?] OF attribute FOR parent_entity;
    uniqueness_rules    : SET[0:?] OF uniqueness_rule FOR parent_entity;
    global_rules        : SET[0:?] OF global_rule FOR entities;
END_ENTITY;
(*
```

Определения атрибутов

supertypes — список типов объектов, для которых тип является непосредственным подтипом, приведенный в алфавитном порядке соответственно значению атрибута **entity_definition.name**. Если тип объекта является результатом отображения ключевых слов ANDOR или AND языка EXPRESS, все дублирующие супертипы удаляются из данного списка;

complex — булевское значение, соответствующее TRUE, если **entity_definition** является результатом отображения супертипов ANDOR или AND в прикладной схеме (см. А.1.3), и FALSE, если **entity_definition** отображен непосредственно из типа объекта в схеме;

instantiable — булевское значение, соответствующее FALSE, если тип объекта объявлен как абстрактный супертип (ABSTRACT SUPERTYPE) в схеме, и TRUE, если нет;

independent — булевское значение, соответствующее FALSE, если тип объекта не является независимым экземпляром, потому что он доступен при помощи спецификации REFERENCE или неявно импортирован в данную схему, и TRUE, если тип объекта объявлен в схеме локально или доступен при помощи спецификации USE;

attributes — атрибуты, объявленные или переобъявленные (см. 9.2.3.4 ГОСТ Р ИСО 10303-11) в типе объекта. Атрибуты, унаследованные из супертипа, не являются элементами данного множества. Поскольку объекты, установленные отображением ключевых слов ANDOR или AND языка EXPRESS, рассматриваются как подтипы составных типов объектов, данное множество является пустым для экземпляров **entity_definition**, установленных отображением ограничения супертипа из ANDOR или AND языка EXPRESS;

uniqueness_rules — правила уникальности, объявленные в типе объекта. Это множество является пустым для объектов, установленных отображением ограничения супертипа из ANDOR или AND языка EXPRESS;

global_rules — глобальные правила, для которых имя типа объекта представлено в объявлении правила.

6.4.13 Объект **attribute**

Объект **attribute** является свойством типа объекта, может быть явным, инверсным или вычисляемым. Атрибут имеет имя и область значений.

EXPRESS-спецификация

```
*)
ENTITY attribute
    ABSTRACT SUPERTYPE OF (ONEOF(derived_attribute, explicit_attribute,
      inverse_attribute));
    name            : express_id;
    parent_entity   : entity_definition;
END_ENTITY;
(*
```

Определения атрибутов

name — имя атрибута;

parent_entity — тип объекта, в котором объявлен атрибут.

6.4.14 Объект **derived_attribute**

Объект **derived_attribute** является атрибутом, значение которому присваивается при вычислении соответствующего выражения. Может переопределять явный или вычисляемый атрибут (см. 9.2.3.4 ГОСТ Р ИСО 10303-11).

EXPRESS-спецификация

```
*)
ENTITY derived_attribute
    SUBTYPE OF (attribute);
    domain          : base_type;
    redeclaring     : OPTIONAL explicit_or_derived;
```

```
END_ENTITY;
(*
```

Определения атрибутов

domain — тип данных результата вычисления значения атрибута;

redeclaring — переопределяемый атрибут (при его наличии).

6.4.15 Объект **explicit_attribute**

Объект **explicit_attribute** является атрибутом, область значения которого явно определена. Может переопределять явный атрибут (см. 9.2.3.4 ГОСТ Р ИСО 10303-11).

EXPRESS-спецификация

```
*)
ENTITY explicit_attribute
    SUBTYPE OF (attribute);
    domain          : base_type;
    redeclaring     : OPTIONAL explicit_attribute;
    optional_flag   : BOOLEAN;
END_ENTITY;
(*
```

Определения атрибутов

domain — тип данных, на который ссылается атрибут;

redeclaring — переопределяемый атрибут (при его наличии);

optional_flag — булевское значение, соответствующее TRUE, если атрибут объявлен необязательным (OPTIONAL), и FALSE, если атрибут не объявлен как необязательный.

6.4.16 Объект **inverse_attribute**

Объект **inverse_attribute** является атрибутом, охватывающим обратные связи отношений, установленных объектом **explicit_attribute**, могущим накладывать на них ограничения и представляться как инверсный (INVERSE) атрибут языка EXPRESS. Может переопределять инверсный атрибут (см. 9.2.3.4 ГОСТ Р ИСО 10303-11). Инверсный атрибут может представляться простым типом объекта или типами SET или BAG. Тип объекта для **inverse_attribute** называется текущим типом объекта, а тип объекта для **explicit_attribute** — типом ссылочного объекта (см. 9.2.1.3 ГОСТ Р ИСО 10303-11).

EXPRESS-спецификация

```
*)
ENTITY inverse_attribute
    SUBTYPE OF (attribute);
    domain          : entity_definition;
    redeclaring     : OPTIONAL inverse_attribute;
    inverted_attr   : explicit_attribute;
    min_cardinality : OPTIONAL bound;
    max_cardinality : OPTIONAL bound;
    duplicates      : BOOLEAN;
END_ENTITY;
(*
```

Определения атрибутов

domain — тип ссылочного объекта, определяющий прямое отношение; источник отношения;

redeclaring — переопределяемый атрибут (при его наличии);

inverted_attr — атрибут в типе ссылочного объекта, связь которого инвертируется;

min_cardinality — минимальное число ссылок (при их наличии) из инвертируемого атрибута в экземпляры типа ссылочного объекта, когда инверсный атрибут представлен операторами BAG или SET. При отсутствии данного атрибута инверсный атрибут представляется единственным типом данных объекта, а не операторами BAG или SET;

max_cardinality — максимальное число ссылок (при их наличии) из инвертируемого атрибута в экземпляры типа ссылочного объекта. При отсутствии данного атрибута оператор BAG или SET, представляющий инверсный атрибут, не определяет максимальное число ссылок или инверсию, представленную типом данных простого объекта;

duplicates — булевское значение, соответствующее TRUE, если инверсный атрибут представлен оператором BAG, и FALSE, если инверсный атрибут представлен оператором SET или типом простого объекта.

6.4.17 Объект **uniqueness_rule**

Объект **uniqueness_rule** представляет правило UNIQUE языка EXPRESS и определяет комбинацию атрибутов, которая должна быть уникальной в рамках экземпляра объекта **entity_definition**, внутри которого объявлено данное правило.

EXPRESS-спецификация

```
*)
ENTITY uniqueness_rule;
    label            : OPTIONAL express_id;
    attributes       : LIST [1:?] OF attribute;
    parent_entity    : entity_definition;
END_ENTITY;
(*
```

Определения атрибутов

label — имя правила уникальности (при его наличии);

attributes — список атрибутов, составляющих правило уникальности;

parent_entity — тип объекта, в котором объявлено данное правило.

6.4.18 Объект **where_rule**

Объект **where_rule** ограничивает совокупность и представляет правило WHERE языка EXPRESS. Когда объект объявлен в **entity_definition** или **defined_type**, или **global_rule**, то соответственно ограничиваются значения атрибутов в типе определяемого объекта или область значений определяемого типа, или значения атрибутов в экземплярах типов объектов, для которых применяется глобальное правило, или само существование этих экземпляров.

EXPRESS-спецификация

```
*)
ENTITY where_rule;
    label            : OPTIONAL express_id;
    parent_item      : type_or_rule;
END_ENTITY;
(*
```

Определения атрибутов

label — имя правила «where» (при его наличии);

parent_item — тип объекта, определенный тип или глобальное правило, в которых объявлено правило «where».

6.4.19 Объект **global_rule**

Объект **global_rule** ограничивает все экземпляры типа объекта или экземпляры типов множественных объектов и представляет оператор RULE языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY global_rule;
    name             : express_id;
    entities         : LIST [1:?] OF entity_definition;
    where_rules      : LIST [1:?] OF where_rule;
    parent_schema    : schema_definition;
END_ENTITY;
(*
```

Определения атрибутов

name — имя правила;

entities — типы объектов, ограниченных данным правилом, определенные в его объявлении в порядке их перечисления в данном объявлении. Объекты, установленные ограничением супертипа в операторах ANDOR или AND языка EXPRESS, не должны присутствовать в данном списке;

where_rules — области значений правил, объявленные в операторе RULE языка EXPRESS в порядке их перечисления в данном операторе;

parent_schema — схема, в которой объявлено правило.

6.4.20 Объект **simple_type**

Объект **simple_type** является неструктурированным, встроенным основным типом языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY simple_type
    ABSTRACT SUPERTYPE OF (ONEOF(integer_type, real_type, string_type,
      binary-type, logical_type, boolean-type, number_type));
END_ENTITY;
(*
```

6.4.21 Объект **number_type**

Объект **number_type** является объектом **simple_type**, представляющим числовой (NUMBER) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY number_type
    SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

6.4.22 Объект **integer_type**

Объект **integer_type** является объектом **simple_type**, представляющим целочисленный (INTEGER) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY integer_type
    SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

6.4.23 Объект **real_type**

Объект **real_type** является объектом **simple_type**, представляющим действительный (REAL) тип языка EXPRESS. В значении действительного типа может быть задано минимальное число значащих цифр.

EXPRESS-спецификация

```
*)
ENTITY real_type
    SUBTYPE OF (simple_type);
    precision : OPTIONAL bound;
END_ENTITY;
(*
```

Определение атрибута

precision — минимальное число значащих цифр в значении типа (при его наличии).

Неформальное утверждение

precision_positive — значение ограничения точности должно приводиться к целому положительному числу (если точность задана).

6.4.24 Объект **string_type**

Объект **string_type** является объектом **simple_type**, представляющим строковый (STRING) тип языка EXPRESS. Строковый тип может иметь заданную фиксированную или переменную ширину (число символов).

EXPRESS-спецификация

```
*)
ENTITY string_type
    SUBTYPE OF (simple_type);
    width          : OPTIONAL bound;
    fixed_width    : BOOLEAN;
END_ENTITY;
(*
```

Определения атрибутов

width — максимальное или, для строк с фиксированной шириной, точное число символов в значении типа (при наличии данного атрибута);

fixed_width — булевское значение, соответствующее TRUE, если тип имеет в качестве области значений строки с фиксированной шириной, и FALSE, если тип имеет в качестве области значений строки с переменной шириной.

Неформальное утверждение

width_positive — значение границы ширины строки должно приводиться к целому положительному числу (если граница задана).

6.4.25 Объект **binary_type**

Объект **binary_type** является объектом **simple_type**, представляющим двоичный (BINARY) тип языка EXPRESS. Двоичный тип может иметь заданную фиксированную или переменную ширину (количество битов).

EXPRESS-спецификация

```
*)
ENTITY binary_type
    SUBTYPE OF (simple_type);
    width          : OPTIONAL bound;
    fixed_width    : BOOLEAN;
END_ENTITY;
(*
```

Определения атрибутов

width — максимальное или, для двоичного числа с фиксированной шириной, точное число битов в значении типа;

fixed_width — булевское значение, соответствующее TRUE, если областью значений типа являются двоичные типы фиксированной ширины, и FALSE, если областью значений типа являются двоичные типы переменной ширины.

Неформальное утверждение

width_positive — значение границы ширины двоичного числа должно приводиться к целому положительному числу (если граница задана).

6.4.26 Объект **logical_type**

Объект **logical_type** является объектом **simple_type**, представляющим логический (LOGICAL) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY logical_type
    SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

6.4.27 Объект **boolean_type**

Объект **boolean_type** является объектом **simple_type**, представляющим булевский (BOOLEAN) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY boolean_type
    SUBTYPE OF (simple_type);
END_ENTITY;
(*
```

6.4.28 Объект **enumeration_type**

Объект **enumeration_type** представляет перечисляемый (ENUMERATION) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY enumeration_type;
    elements : LIST [1:?] OF UNIQUE express_id;
END_ENTITY;
(*
```

Определение атрибута

elements — список значений типов в порядке их перечисления в объявлении перечисляемого типа языка EXPRESS.

6.4.29 Объект **select_type**

Объект **select_type** представляет выбираемый (SELECT) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY select_type;
    selections : SET [1:?] OF named_type;
END_ENTITY;
(*
```

Определение атрибута

selections — множество выбираемых типов.

6.4.30 Объект **aggregation_type**

Объект **aggregation_type** является типом данных языка EXPRESS, значения которого представляют наборы (коллекции) других значений заданного основного типа.

EXPRESS-спецификация

```
*)
ENTITY aggregation_type
    ABSTRACT SUPERTYPE OF (ONEOF(variable_size_aggregation_type,
      array_type));
    element_type : base_type;
END_ENTITY;
(*
```

Определение атрибута

element_type — тип элементов, которые содержатся в значениях агрегатного типа.

6.4.31 Объект **variable_size_aggregation_type**

Объект **variable_size_aggregation_type** является объектом **aggregation_type**, объявленным как имеющий переменное число элементов. Число элементов ограничено снизу и может быть ограничено сверху.

EXPRESS-спецификация

```
*)
ENTITY variable_size_aggregation_type
    ABSTRACT SUPERTYPE OF (ONEOF(set_type, bag_type, list_type))
    SUBTYPE OF (aggregation_type);
    lower_bound : bound;
    upper_bound : OPTIONAL bound;
END_ENTITY;
(*
```

Определения атрибутов

lower_bound — минимальное число элементов, могущее присутствовать в экземпляре типа. Нижнее значение границы равно нулю, если отсутствует числовое выражение, определяющее данную величину в схеме, в которой объявлен агрегат;

upper_bound — максимальное число элементов (при его наличии), могущее присутствовать в экземпляре типа. Если оно отсутствует, число элементов в экземпляре типа сверху не ограничено.

Неформальное утверждение

valid_boundaries — значение нижней границы не может быть больше значения верхней.

6.4.32 Объект **set_type**

Объект **set_type** является объектом **variable_size_aggregation_type**, представляющим выбираемый (SET) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY set_type
    SUBTYPE OF (variable_size_aggregation_type);
END_ENTITY;
(*
```

6.4.33 Объект **bag_type**

Объект **bag_type** является объектом **variable_size_aggregation_type**, представляющим мультимножественный (BAG) тип языка EXPRESS.

EXPRESS-спецификация

```
*)
ENTITY bag_type
    SUBTYPE OF (variable_size_aggregation_type);
END_ENTITY;
(*
```

6.4.34 Объект **list_type**

Объект **list_type** является объектом **variable_size_aggregation_type**, представляющим списочный (LIST) тип языка EXPRESS. Может быть потребована уникальность элементов списка.

EXPRESS-спецификация

```
*)
ENTITY list_type
    SUBTYPE OF (variable_size_aggregation_type);
    unique_flag : BOOLEAN;
END_ENTITY;
(*
```

Определение атрибута

unique_flag — булевское значение, соответствующее TRUE, если ключевое слово UNIQUE установлено в определении объекта **list_type** данной схемы, FALSE — в противном случае.

6.4.35 Объект **array_type**

Объект **array_type** является объектом **aggregation_type**, представляющим тип массива (ARRAY) языка EXPRESS. Массив имеет значения нижнего и верхнего индексов. При наличии ключевого слова UNIQUE элементы массива должны быть уникальными. При наличии ключевого слова OPTIONAL в одной или нескольких индексированных позициях массив может содержать неопределенные значения.

EXPRESS-спецификация

```
*)
ENTITY array_type
    SUBTYPE OF (aggregation_type);
    lower_index     : bound;
    upper_index     : bound;
    unique_flag     : BOOLEAN;
    optional_flag   : BOOLEAN;
END_ENTITY;
(*
```

Определения атрибутов

lower_index — самый нижний значащий индекс для экземпляров типа;

upper_index — самый верхний значащий индекс для экземпляров типа;

unique_flag — булевское значение, соответствующее TRUE, если ключевое слово UNIQUE установлено в определении объекта **array_type** данной схемы, FALSE — в противном случае;

optional_flag — булевское значение, соответствующее TRUE, если ключевое слово OPTIONAL установлено в определении объекта **array_type** данной схемы, FALSE — в противном случае.

Неформальное утверждение

valid_boundaries — значение нижней границы индекса не может быть больше значения верхней границы индекса.

6.4.36 Объект **bound**

Объект **bound** является ограничением агрегатного, двоичного, строкового и действительного типов языка EXPRESS, установленным в числовом выражении, имеющем целочисленное значение. Значение **bound** может основываться исключительно на схеме, в которой оно объявлено, или зависеть от совокупности данной схемы.

EXPRESS-спецификация

```
*)
ENTITY bound
    ABSTRACT SUPERTYPE OF (ONEOF(integer_bound,
      population_dependent_bound));
END_ENTITY;
(*
```

6.4.37 Объект **population_dependent_bound**

Объект **population_dependent_bound** является объектом **bound**, значение которого зависит от совокупности схемы, в которой он объявлен.

EXPRESS-спецификация

```
*)
ENTITY population_dependent_bound
    SUBTYPE OF (bound);
END_ENTITY;
(*
```

6.4.38 Объект **integer_bound**

Объект **integer_bound** является объектом **bound**, значение которого базируется исключительно на схеме, внутри которой он объявлен.

EXPRESS-спецификация

```
*)
ENTITY integer_bound
    SUBTYPE OF (bound);
    bound_value : INTEGER;
END_ENTITY;
(*
```

Определение атрибута

bound_value — целочисленное значение границы.

```
*)
END_SCHEMA;     --SDAI_dictionary_schema
(*
```

7 Схема сеанса СИДД

Следующее объявление на языке EXPRESS начинает схему сеанса СИДД и определяет необходимые внешние ссылки.

EXPRESS-спецификация

```
*)
SCHEMA SDAI_session_schema;
REFERENCE FROM SDAI_parameter_data_schema
    (entity_instance, aggregate_instance);
USE FROM SDAI_population_schema;
(*
```

Примечание — Схемы, упомянутые выше, можно найти в следующих разделах настоящего стандарта:

SDAI_parameter_data_schema — в разделе 9;

SDAI_population_schema — в разделе 8.

7.1 Введение

Схема сеанса СИДД определяет структуру данных, необходимых для управления сеансом. Текущее состояние сеанса СИДД и его взаимодействия с реализацией СИДД, такие как режимы доступа, сообщения (транзакции), хранилища и ошибки сеанса, делаются доступными через совокупность схемы сеанса СИДД. Как и во всех схемах, доступных в словаре данных, элементы, импортируемые в схему сеанса СИДД из схемы совокупности СИДД, схемы параметризованных данных и словаря СИДД через схему совокупности СИДД, должны быть разрешены в схеме сеанса СИДД (см. А.1.1).

Схема сеанса СИДД описывает отдельное приложение, отдельное представление пользователя информации о сеансе СИДД. Реализация СИДД должна создавать экземпляры типов данных объектов схемы сеанса СИДД, не импортированных из другой схемы, доступные в отдельной СИДД-модели. Данная СИДД-модель должна быть связана с одним экземпляром схемы. И СИДД-модель, и экземпляр схемы должны базироваться на схеме сеанса СИДД. Объектом **sdai_model.name** этой СИДД-модели должен быть ‘SDAI_SESSION_SCHEMA_DATA’. Объектом **schema_instance.name** этого

экземпляра схемы должен быть 'SDAI_SESSION_SCHEMA_INSTANCE'. Данные СИДД-модель и экземпляр схемы не должны продолжать существование после окончания сеанса.

Экземпляры объектов схемы сеанса СИДД создаются и изменяются только вследствие конкретных операторов СИДД, а не операторов объектов или прикладного экземпляра.

Пример 6 — Операция открытия сеанса создает экземпляры типа объекта сеанса и объекта реализации.

7.2 Фундаментальные принципы и допущения

Структура объектов и атрибутов схемы сеанса СИДД учитывает реализацию и эффективность доступа.

7.3 Определения типов схемы сеанса СИДД

В данном подразделе описаны принципы сеанса, определенные конструкцией TYPE языка EXPRESS.

7.3.1 Тип **access_type**

Тип **access_type** определяет режимы доступа «только чтение» или «чтение — запись» для **sdai_transaction** или **sdai_model**.

EXPRESS-спецификация

```
*)
TYPE access_type = ENUMERATION OF
      (read_only,
      read_write);
END_TYPE;
(*
```

Элементы перечисления

read_only — значение, указывающее доступ в режиме «только чтение»;

read_write — значение, указывающее доступ в режиме «чтение — запись».

7.3.2 Тип **error_base**

Тип **error_base** является выбором между **entity_instance** (экземпляром объекта) или **aggregate_instance** (экземпляром агрегата) и связан с конкретной ошибкой, выданной реализацией СИДД.

EXPRESS-спецификация

```
*)
TYPE error_base = SELECT
      (entity_instance,
      aggregate_instance);
END_TYPE;
(*
```

7.3.3 Тип **time_stamp**

Тип **time_stamp** является спецификацией даты и времени. Содержание строки должно соответствовать расширенному формату полной календарной даты, установленному в 5.2.1.1 ИСО 8601, объединенному с расширенным форматом календарного времени, установленным в 5.3.1.1 или 5.3.3 ИСО 8601. Дата и время должны быть разделены заглавной буквой Т согласно 5.4.1.1 ИСО 8601. Альтернативные форматы из 5.3.1.1 и 5.3.3 ИСО 8601 разрешают необязательное включение спецификатора временно́й зоны.

EXPRESS-спецификация

```
*)
TYPE time_stamp = STRING(256);
END_TYPE;
(*
```

7.4 Определения объектов схемы сеанса СИДД

В данном подразделе устанавливаются понятия сеанса, определяющие применение конструкции ENTITY языка EXPRESS.

7.4.1 Объект **sdai_session**

Объект **sdai_session** представляет информацию, описывающую сеанс СИДД во время активной реализации СИДД. Содержит информацию, отражающую состояние сеанса и относящуюся к сообщениям (транзакциям), ошибкам, записям событий, хранилищам и словарю данных.

EXPRESS-спецификация

```
*)
ENTITY sdai_session;
    sdai_implementation  : implementation;
    recording_active     : BOOLEAN;
    errors               : LIST [0:?] OF error-event;
    known_servers        : SET [1:?] OF sdai_repository;
    active_servers       : SET [1:?] OF sdai_repository;
    active_models        : SET [1:?] OF sdai_model;
    data_dictionary      : OPTIONAL schema_instance;
INVERSE
    active_transaction   : SET [0:1] OF sdai_transaction FOR owning_session;
END_ENTITY;
(*
```

Определения атрибутов

sdai_implementation — характеристики реализации СИДД;

recording_active — булевское значение, соответствующее FALSE, если запись событий запрещена, TRUE — в противном случае;

errors — список ошибок, произошедших в результате выполнения предыдущих команд СИДД, пока запись событий была активна;

known_servers — хранилища, доступные приложению в данном сеансе. Наличие конкретных хранилищ зависит от специфической установки реализации СИДД;

active_servers — хранилища, открытые в данном сеансе;

active_models — множество **sdai_models**, доступных в данном сеансе;

data_dictionary — экземпляр схемы (при его наличии), основанный на схеме сеанса СИДД, с которой связаны СИДД-модели, содержащие словарь данных. Для реализаций СИДД, соответствующих соглашениям по классам реализации 2—6, данный атрибут должен иметь установленное значение;

active_transaction — транзакция (сообщение), обеспечивающее доступ к СИДД-моделям и экземплярам схем в данном сеансе.

7.4.2 Объект **implementation**

Объект **implementation** представляет программный продукт, обеспечивающий функциональные возможности, определенные языковой привязкой СИДД.

EXPRESS-спецификация

```
*)
ENTITY implementation;
    name                    : STRING;
    level                   : STRING;
    sdai_version            : STRING;
    binding_version         : STRING;
    implementation_class    : INTEGER;
    transaction_level       : INTEGER;
    expression_level        : INTEGER;
    recording_level         : INTEGER;
    scope_level             : INTEGER;
    domain_equivalence_level : INTEGER;
END_ENTITY;
(*
```

Определения атрибутов

name — имя объекта **implementation**, присвоенное разработчиком;

level — уровень версии программного средства для объекта **implementation**, определенный разработчиком;

sdai_version — версия (редакция) настоящего стандарта, которой соответствует данная реализация. Значение данного атрибута должно удовлетворять методу регистрации, установленному в 4.3 ГОСТ Р ИСО 10303-1, и быть идентификатором объекта для соответствующей версии настоящего стандарта (см. С.1);

binding_version — версия языковой привязки СИДД, поддерживаемая в соответствии с определением языковой привязки СИДД;

implementation_class — класс реализации, установленный в настоящем стандарте, которому соответствует объект **implementation**;

transaction_level — уровень транзакции (сообщения), поддерживаемый реализацией (см. 13.1.1);

expression_level — уровень вычисления выражений, поддерживаемый реализацией (см. 13.1.2);

recording_level — уровень отчета (записи) о событии, поддерживаемый реализацией (см. 13.1.3);

scope_level — уровень области действия (применения), поддерживаемый реализацией (см. 13.1.4);

domain_equivalence_level — уровень эквивалентности области значений, поддерживаемый реализацией (см. 13.1.5).

Неформальные утверждения

class1to7 — атрибут **implementation_class** должен иметь значения 1—7, которые должны соответствовать классу реализации, установленному в 13.2;

trans1to3 — атрибут **transaction_level** должен иметь значения 1—3, которые должны соответствовать уровню транзакции, установленному в 13.1.1;

expr1to4 — атрибут **expression_level** должен иметь значения 1—4, которые должны соответствовать уровню вычисления выражения, установленному в 13.1.2;

rec1to2 — атрибут **recording_level** должен иметь значение 1 или 2, которое должно соответствовать уровню отчета о событии, установленному в 13.1.3;

scope1to2 — атрибут **scope_level** должен иметь значение 1 или 2, которое должно соответствовать уровню области действия, установленному в 13.1.4;

equiv1to2 — атрибут **domain_equivalence_level** должен иметь значение 1 или 2, которое должно соответствовать уровню поддержки SCOPE, установленному в 13.1.5.

7.4.3 Объект **sdai_repository**

Объект **sdai_repository** представляет идентификацию средства, посредством которого **sdai_models** и **schema_instances** могут быть сохранены в течение сеанса.

Примечание — Данный объект предназначен для обеспечения физического размещения СИДД-моделей и экземпляров схем.

EXPRESS-спецификация

```
*)
ENTITY sdai_repository;
    name          : STRING;
    contents      : sdai_repository_contents;
    description   : STRING;
INVERSE
    session : sdai_session FOR known_servers;
UNIQUE
    UR1 : name, session;
END_ENTITY;
(*
```

Определения атрибутов

name — имя объекта **sdai_repository**. Данное имя чувствительно к регистру;

contents — имеющиеся в хранилище СИДД-модели и экземпляры схем;

description — описание хранилища;

session — текущий сеанс.

Формальное утверждение

UR1 — имя должно быть уникальным в текущем сеансе.

7.4.4 Объект **sdai_repository_contents**

Объект **sdai_repository_contents** идентифицирует объекты **sdai_models** и **schema_instances**, имеющиеся в хранилище.

EXPRESS-спецификация

```
*)
ENTITY sdai_repository_contents;
    models   : SET [0:?] OF sdai_model;
    schemas  : SET [0:?] OF schema_instance;
```

```
INVERSE
    repository : sdai_repository FOR contents;
END_ENTITY;
(*
```

Определения атрибутов

models — набор СИДД-моделей в хранилище;

schemas — набор экземпляров схем в хранилище;

repository — хранилище, содержащее СИДД-модели и экземпляры схем.

7.4.5 Объект **sdai_transaction**

Объект **sdai_transaction** описывает в возможность текущего доступа к данным (RW или RO) во время сеанса. Транзакции (сообщения) не могут существовать вне сеанса, и только одна транзакция должна быть активна в любой заданный момент времени. Транзакции требуются только для реализаций СИДД, поддерживающих третий уровень транзакции (см. 13.1.1).

EXPRESS-спецификация

```
*)
ENTITY sdai_transaction;
    mode            : access_type;
    owning_session  : sdai_session;
END_ENTITY;
(*
```

Определения атрибутов

mode — доступ в режимах «только чтение» или «чтение — запись», обеспечиваемые транзакцией в сеансе СИДД (**sdai_session**);

owning_session — сеанс СИДД (**sdai_session**), в котором транзакция активна.

7.4.6 Объект **event**

Объект **event** является нотационной записью некоторого события, связанного с командой СИДД в некоторый момент времени во время сеанса.

Примечание — Последующие версии (редакции) СИДД могут расширить перечень типов событий, которые могут быть описаны помимо отчетов об ошибках, установленных в настоящем стандарте.

EXPRESS-спецификация

```
*)
ENTITY event
ABSTRACT SUPERTYPE OF (error_event);
function_id     : STRING;
time            : time_stamp;
END_ENTITY;
(*
```

Определения атрибутов

function_id — идентификатор функции или подпрограммы СИДД, с которой связано событие. Значения данного атрибута и их соответствие командам СИДД определены в языковых привязках СИДД;

time — временная отметка, указывающая момент наступления события.

7.4.7 Объект **error_event**

Объект **error_event** является событием (**event**), сгенерированным в результате неправильно выполненной команды СИДД или полученным как результат команды отчета об ошибке.

EXPRESS-спецификация

```
*)
ENTITY error_event
SUBTYPE OF (event);
error           : INTEGER;
description     : OPTIONAL STRING;
base            : OPTIONAL error_base;
END_ENTITY;
(*
```

Определения атрибутов
error — код ошибки (см. раздел 11);
description — описание обнаруженной ошибки (при ее наличии);
base — экземпляр, связанный с ошибкой или породивший ее (при ее наличии) (см. раздел 11).
*)
END_SCHEMA; -- SDAI_session_schema
(*

8 Схема совокупности СИДД

Следующее объявление на языке EXPRESS начинает схему совокупности СИДД и определяет необходимые внешние ссылки.

EXPRESS-спецификация

```
*)
SCHEMA SDAI_population_schema;
REFERENCE FROM SDAI_parameter_data_schema
(entity_instance,
application_instance);
REFERENCE FROM SDAI_session_schema
(sdai_session,
sdai_repository,
access_type,
time_stamp);
(*
```

В случаях, когда реализация объявляется соответствующей классам реализации 2—6 или обеспечивает доступ к словарю данных СИДД, в схему совокупности СИДД должно быть включено последующее объявление на языке EXPRESS. В противном случае в схему совокупности СИДД должны быть включены объявления на языке EXPRESS из 8.3.

EXPRESS-спецификация

```
*)
USE FROM SDAI_dictionary_schema;
(*
```

Примечание — Схемы, на которые выше даны ссылки, можно найти в следующих разделах настоящего стандарта:

SDAI_parameter_data_schema — раздел 9;
SDAI_session_schema — раздел 7;
SDAI_dictionary_schema — раздел 6.

8.1 Введение

Схема совокупности СИДД определяет структуру для организации, создания и управления экземплярами EXPRESS-объектов. Данная схема импортируется схемой сеанса СИДД, а все элементы данной схемы разрешены на схеме сеанса СИДД в совокупности схемы словаря СИДД (см. А.1.1).

Реализации, не обеспечивающие доступ к словарю данных, должны использовать схему совокупности СИДД без импортирования схемы словаря СИДД. Ссылки на элементы, определенные в схеме словаря СИДД, заменяются строковыми значениями.

Экземпляры типов данных объектов, определенные в схеме совокупности СИДД, обеспечивают структуру управления экземплярами объектов прикладных схем и схемы словаря СИДД. Данная структура управления не требует существования экземпляров типов данных объектов, определенных в самой схеме совокупности СИДД (то есть, экземпляры **sdai_model** не обязаны присутствовать в СИДД-модели). Механизм, обеспечивающий доступность экземпляров типов объектов, объявленных в схеме совокупности СИДД, оставлен на усмотрение реализации.

8.2 Фундаментальные принципы и допущения

Структура объектов и атрибутов схемы сеанса СИДД учитывает реализацию и эффективность доступа.

8.3 Определения типов схемы совокупности СИДД

В данном подразделе содержатся описания концепций, определяющих применение конструкции TYPE языка EXPRESS. В случае, когда реализация не поддерживает доступ к словарю данных, объявления на языке EXPRESS из данного подраздела должны быть включены в схему совокупности СИДД. В противном случае эти объявления не должны быть включены в схему совокупности СИДД.

8.3.1 Тип **schema_definition**

Тип **schema_definition** представляет собой понятие, определенное и представленное атрибутом **schema_definition.name** в 6.4.1.

EXPRESS-спецификация

```
*)
TYPE schema_definition = STRING;
END_TYPE;
(*
```

8.3.2 Тип **entity_definition**

Тип **entity_definition** представляет собой понятие, определенное и представленное атрибутом **entity_definition.name** в 6.4.12.

EXPRESS-спецификация

```
*)
TYPE entity_definition = STRING;
END_TYPE;
(*
```

8.4 Определения объектов схемы совокупности СИДД

В данном подразделе описаны концепции совокупности, определяющие использование конструкции ENTITY языка EXPRESS.

8.4.1 Объект **schema_instance**

Объект **schema_instance** является логическим набором (коллекцией) СИДД-моделей (**sdai_models**). Используется как области значений для проверки глобального правила, обеспечения ссылок между экземплярами объектов в различных СИДД-моделях и проверки правила уникальности. Объект **schema_instance** (экземпляр схемы) базируется на одной схеме. Должно обеспечиваться связывание СИДД-моделей с **schema_instance**, когда СИДД-модели базируются на той же схеме, что и данный **schema_instance**. Также должно обеспечиваться связывание СИДД-моделей с **schema_instance**, когда СИДД-модели базируются на другой схеме, если схема, на которой базируются СИДД-модели, содержит конструкции, объявленные эквивалентными по области значений с конструкциями в схеме, на которой базируется **schema_instance**. Хотя **schema_instance** существует в одном хранилище, должно поддерживаться связывание с СИДД-моделями из любого другого хранилища с данными **schema_instance**.

EXPRESS-спецификация

```
*)
ENTITY schema_instance;
    name                : STRING;
    associated_models   : SET [0:?] OF sdai_model;
    native_schema       : schema_definition;
    repository          : sdai_repository;
    change_date         : OPTIONAL time_stamp;
    validation_date     : time_stamp;
    validation_result   : LOGICAL;
    validation_level    : INTEGER;
UNIQUE
    UR1 : name, repository;
WHERE
    WR1 : SELF IN SELF.repository.contents.schemas;
END_ENTITY;
(*
```

Определения атрибутов

name — имя экземпляра схемы (**schema_instance**), зависящее от регистра символов;

contents — СИДД-модели, связанные с экземпляром схемы;

native_schema — схема, на которой базируется данный экземпляр схемы;

repository — хранилище, в котором создан данный экземпляр схемы;

change_date — дата создания, последнего добавления или удаления СИДД-модели из текущего экземпляра схемы (при ее наличии);

validation_date — дата последнего применения операции проверки текущего экземпляра схемы;

validation_result — результат последнего применения операции проверки текущего экземпляра схемы;

validation_level — уровень вычисления выражения для проверки правильности реализации, выполнявшей последнюю команду проверки правильности экземпляра схемы для текущего экземпляра схемы (см. 13.1.2).

Формальное утверждение

UR1 — имя должно быть уникальным в хранилище, содержащем экземпляр схемы.

Неформальное утверждение

dictionary instance — во время сеанса СИДД для реализации, поддерживающей словарь данных СИДД, должен существовать экземпляр **schema_instance**, с которым будут связаны все СИДД-модели, составляющие словарь данных СИДД.

8.4.2 Объект **sdai_model**

Объект **sdai_model** является механизмом группирования, содержащим набор связанных экземпляров объектов, основанных на **schema_definition**.

Примечание — Связь экземпляров объектов, сгруппированных в СИДД-модель, не определена. Однако предполагается, что группировка основана на некоторой логической связи между экземплярами объектов, которая является полезной для прикладной программы при управлении прикладными данными.

EXPRESS-спецификация

```
*)
ENTITY sdai_model;
    name                    : STRING;
    contents                : sdai_model_contents;
    underlying_schema       : schema_definition;
    repository              : sdai_repository;
    change_date             : OPTIONAL time_stamp;
    mode                    : OPTIONAL access_type;
INVERSE
    associated_with : SET [0:?] OF schema_instance FOR associated_models;
UNIQUE
    UR1 : repository, name;
WHERE
    WR1 : SELF IN SELF.repository.contents.models;
END_ENTITY;
(*
```

Определения атрибутов

name — идентификатор для данной **sdai_model**. Данное имя чувствительно к регистру;

contents — механизм объединения экземпляров объектов внутри **sdai_model**;

underlying_schema — схема, определяющая структуру данных, появляющихся в СИДД-модели;

repository — хранилище, внутри которого создана СИДД-модель;

change_date — дата создания или последнего изменения, включая создание или удаление экземпляра объекта внутри текущей СИДД-модели (при ее наличии);

mode — текущий режим доступа к **sdai_model** (при его наличии). Если он отсутствует, **sdai_model** не открыта;

associated_with — экземпляры схем, с которыми связана СИДД-модель.

Формальное утверждение

UR1 — имя должно быть уникально внутри хранилища, содержащего СИДД-модель.

8.4.3 Объект **sdai_model_contents**

Объект **sdai_model_contents** содержит экземпляры объектов, составляющих **sdai_model**. Экземпляры объектов доступны в виде единого набора (коллекции), независимо от типа данных объекта и сгруппированы в поднаборы по типу данных объекта.

Примечание — Доступность и группировка экземпляров объектов по трем альтернативным атрибутам являются единственным случаем, когда в данной схеме учитывается легкость доступа к ним со стороны прикладного программиста.

EXPRESS-спецификация

```
*)
ENTITY sdai_model_contents;
    instances            : SET [0:?] OF entity_instance;
    folders              : SET [0:?] OF entity_extent;
    populated_folders    : SET [0:?] OF entity_extent;
END_ENTITY;
(*
```

Определения атрибутов

instances — набор всех экземпляров объектов в **sdai_model**, независимо от типа данных объекта;

folders — набор **entity_extents** для всех типов объектов, доступных в схеме, соответствующий СИДД-модели. Содержит по одному члену для каждого объекта **entity_definition**, установленного в схеме, управляющей СИДД-моделью, независимо от текущего существования экземпляров объектов данного типа;

populated_folders — поднабор **folders**, содержащий набор **entity_extents**, для которых в данный момент в СИДД-модели существуют экземпляры объекта.

Неформальное утверждение

IP1 — набор **sdai_model_contents.instances**, содержащий те же экземпляры объектов, что и объединение набора расширений содержимого **sdai_model_contents.populated_folders**.

8.4.4 Объект **entity_extent**

Объект **entity_extent** группирует все экземпляры типов данных объекта, существующих в **sdai_model**. Это группирование включает в себя экземпляры определенного **entity_definition**, всех подтипов данного **entity_definition** и других **entity_definition**, полученных в результате отображения конструкций AND и ANDOR языка EXPRESS, описанного в приложении А и содержащего тип данных объекта в качестве составной части.

Примечание — Такое группирование экземпляров конкретного типа полезно в качестве точки входа внутрь СИДД-модели для получения доступа к экземплярам любых типов корневых объектов в схемах, содержащих древовидные или иерархические структуры.

EXPRESS-спецификация

```
*)
ENTITY entity_extent;
    definition   : entity_definition;
    instances    : SET [0:?] OF entity_instance;
INVERSE
    owned_by     : sdai_model_contents FOR folders;
END_ENTITY;
(*
```

Определения атрибутов

definition — объект **entity_definition**, экземпляры которого содержатся в папке;

instances — экземпляры объектов, содержащиеся в данной папке;

owned_by — содержимое СИДД-модели, к которой принадлежит данный **entity_extent.**

8.4.5 Объект **scope**

Объект **scope** создает структуру, обеспечивающую область действия (применения) ссылок и существующих отношений между экземплярами объектов, определяемыми в конструкции SCOPE согласно 10.3 ГОСТ Р ИСО 10303-21. Настоящий стандарт ссылается на прикладной экземпляр, внутри которого определена структура **scope**, являющаяся прерогативой владельца данной области. Прикладные экземпляры, принадлежащие **scope**, могут иметь собственную область действия (**scope**). Объект **scope** существует в той же СИДД-модели, что и прикладной экземпляр (**application_instance**), являющийся владельцем данной области (например, если СИДД-модель, содержащая прикладной экземпляр, удаляется, то его область действия (**scope**) также удаляется).

EXPRESS-спецификация

```
*)
```

```
ENTITY scope;
    owner          : application_instance;
    owned          : SET [1:?] OF application_instance;
    export_list    : SET [0:?] OF application_instance;
UNIQUE
    UR1 : owner;
WHERE
    WR2 : NOT (owner IN owned);
END_ENTITY;
(*
```

Определения атрибутов

owner — прикладной экземпляр, в котором определена текущая область действия (**scope**);

owned — прикладные экземпляры, определенные внутри текущей **scope**;

export_list — прикладные экземпляры, экспортируемые из данной области действия (**scope**), на которые возможно ссылаться.

Формальные утверждения

UR1 — прикладной экземпляр не может обладать более чем одной областью действия (**scope**);

WR2 — прикладной экземпляр не может быть включен в набор прикладных экземпляров, которыми он обладает.

Неформальные утверждения

IP1 — структура SCOPE должна быть ациклической;

IP2— каждый экспортируемый прикладной экземпляр должен быть членом набора собственных прикладных экземпляров или членом экспортного списка (**export_list**) одного экземпляра области действия, владельцем которой является один из членов данного списка.

```
*)
END_SCHEMA; -- SDAI_population_schema
(*
```

9 Схема параметризированных данных СИДД

Следующее объявление на языке EXPRESS открывает схему параметризированных данных СИДД и определяет необходимые внешние ссылки.

EXPRESS-спецификация

```
*)
SCHEMA SDAI_parameter_data_schema;
REFERENCE FROM SDAI_population_schema
    (schema_instance,
    sdai_model,
    sdai_model_contents);
REFERENCE FROM SDAI_session_schema
    (sdai_repository);
(*
```

П р и м е ч а н и е — Схемы, на которые выше даны ссылки, можно найти в следующих разделах настоящего стандарта:

SDAI_session_schema — раздел 7;
SDAI_population_schema — раздел 8.

9.1 Введение

Схема параметризированных данных СИДД содержит концептуальные описания данных, передаваемых как параметры или обрабатываемых через СИДД. В отличие от схем словаря, сеанса и совокупности СИДД, данная схема не требует реализации и не должна быть частью словаря данных СИДД. Данная схема определена для поддержки описания команд СИДД и определения среды СИДД, в которой существуют экземпляры объектов. В отдельных случаях в данном разделе отсутствуют обязательные EXPRESS-спецификации атрибутов объекта, объявленного в схеме парамет-

ризированных данных СИДД. Вместо этого EXPRESS-спецификации данных атрибутов приведены в примере, а в основной части стандарта приведено только текстовое описание атрибута, так как обеспечение описания параметра важнее, чем определение его реализации.

Схема параметризированных данных СИДД описывает отношения подтипов между экземплярами типов данных объекта, управляемых посредством команд СИДД. Данные отношения подтипов используют для классификации команд СИДД. Эта классификация позволяет определить поведение всех реализаций СИДД по отношению к типам экземпляров объекта, которые могут быть использованы в качестве параметров или возвращаемых значений для команд СИДД.

Языковые привязки СИДД определяют типы данных для **entity_instance** и его подтипы. Реализации СИДД должны обеспечивать типы данных этих объектов, поведение которых должно соответствовать языковым реализациям отношений подтипов, установленным в данной схеме. Тип данных, представляющий тип объекта **application_instance**, должен вести себя как супертип для каждого типа объекта, объявленного в прикладной схеме. Тип данных, представляющий тип объекта **dictionary_instance**, должен вести себя как супертип для каждого типа объекта, объявленного в схеме словаря СИДД. Тип данных, представляющий тип объекта **session_instance**, должен вести себя как супертип для каждого типа объекта, объявленного в схеме сеанса СИДД.

Пример 7 — Тип данных объекта «e1», заданный прикладной схемой, должен иметь **application_instance** из схемы типа данных СИДД в качестве непосредственного супертипа. Тип данных объекта «e2» не нуждается в **application_instance** в качестве непосредственного супертипа, поскольку он наследуется через «e1».

```
SCHEMA application l;
ENTITY e1;
END_ENTITY;
ENTITY e2;
    SUBTYPE OF (e1);
END_ENTITY;
END_SCHEMA;
```

Отношения подтипа, определенные в данном подразделе, не должны проявляться в совокупности схемы словаря СИДД для любой прикладной схемы, схемы словаря или схемы сеанса СИДД.

9.2 Фундаментальные принципы и допущения

Данная схема допускает, что программное обеспечение реализует представления целочисленных, действительных, символьных, битовых и логических типов данных.

9.3 Определения типов схемы параметризованных данных СИДД

В данном подразделе описаны понятия параметризованных данных, определяющие использование конструкции TYPE языка EXPRESS.

9.3.1 Тип **primitive**

Тип **primitive** является выбором из типов **aggregate_primitive** или **assignable_primitive**. Значением данного типа может быть представление атрибута экземпляра объекта на языке EXPRESS.

EXPRESS-спецификация

```
*)
TYPE primitive = SELECT
    (aggregate_primitive,
    assignable_primitive);
END_TYPE;
(*
```

9.3.2 Тип **assignable_primitive**

Тип **assignable_primitive** является выбором одного из типов **entity_instance**, **string_value**, **binary_value**, **integer_value**, **number_value**, **enumeration_value**, **select_value**, **real_value**, **boolean_value** или **logical_value**, а также является значением, которое может быть напрямую присвоено атрибуту экземпляра объекта на языке EXPRESS посредством команды СИДД.

EXPRESS-спецификация

```
*)
TYPE assignable_primitive = SELECT
    (entity_instance,
    string_value,
    binary_value,
```

```
        integer_value,
        number_value,
        enumeration_value,
        select_value,
        real_value,
        boolean_value,
        logical_value);
END_TYPE;
(*
```

9.3.3 Тип **aggregate_primitive**

Тип **aggregate_primitive** является одним из типов **aggregate_instance** или **select_aggregate_instance**, а также значением исходного типа экземпляра агрегата.

EXPRESS-спецификация

```
*)
TYPE aggregate_primitive = SELECT
        (aggregate_instance,
        select_aggregate_instance);
END_TYPE;
(*
```

9.3.4 Тип **string_value**

Тип **string_value** является значением, связанным с типом данных, определяемым конструкцией STRING языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE string_value = STRING;
END_TYPE;
(*
```

9.3.5 Тип **binary_value**

Тип **binary_value** является значением, связанным с типом данных, определенным конструкцией BINARY языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE binary_value = BINARY;
END_TYPE;
(*
```

9.3.6 Тип **integer_value**

Тип **integer_value** является значением, связанным с типом данных, определенным конструкцией INTEGER языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE integer_value = INTEGER;
END_TYPE;
(*
```

9.3.7 Тип **real_value**

Тип **real_value** является значением, связанным с типом данных, определенным конструкцией REAL языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE real_value = REAL;
END_TYPE;
(*
```

9.3.8 Тип **number_value**

Тип **number_value** является значением, связанным с типом данных, определенным конструкцией NUMBER языка EXPRESS.

EXPRESS-спецификация

```
*)
```

```
TYPE number_value = SELECT
      (real_value,
      integer_value);
END_TYPE;
(*
```

9.3.9 Тип **boolean_value**

Тип **boolean_value** является значением, связанным с типом данных, определенным конструкцией BOOLEAN языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE boolean_value = BOOLEAN;
END_TYPE;
(*
```

9.3.10 Тип **logical_value**

Тип **logical_value** является значением, связанным с типом данных, определенным конструкцией LOGICAL языка EXPRESS.

EXPRESS-спецификация

```
*)
TYPE logical_value = LOGICAL;
END_TYPE;
(*
```

9.3.11 Тип **bound_instance_value**

Тип **bound_instance_value** является значением **bound**, имеющим текущее значение. Текущей величиной является **integer_bound.bound_value** или значение **population_dependent_bound**, когда совокупность схемы, объявляющая **population_dependent_bound**, достаточна для вычисления целочисленного выражения, определяющего данное значение.

EXPRESS-спецификация

```
*)
TYPE bound_instance_value = INTEGER;
END_TYPE;
(*
```

9.3.12 Тип **query_source**

Тип **query_source** является одним из типов **aggregate_instance, sdai_model, sdai_repository** или **schema_instance,** а также областью значений для выполнения команды запроса СИДД.

EXPRESS-спецификация

```
*)
TYPE query_source = SELECT
      (aggregate_instance,
      sdai_model,
      sdai_repository,
      schema_instance);
END_TYPE;
(*
```

9.4 Определения объектов схемы параметризированных данных СИДД

Данный подраздел содержит понятия параметризованных данных, определяющих использование конструкции ENTITY языка EXPRESS.

9.4.1 Объект **iterator**

Объект **iterator** является механизмом, с помощью которого команды СИДД ссылаются на элементы агрегатов. Для агрегата можно определить несколько **iterator**. При этом каждый из **iterator** работает независимо от других; поведение одного **iterator** после изменения агрегата, использующего другой **iterator**, в настоящем стандарте не определено.

EXPRESS-спецификация

```
*)
ENTITY iterator;
      subject              : aggregate_instance;
      current_member       : OPTIONAL primitive;
```

END_ENTITY;
(*
Определения атрибутов
subject — агрегат, доступный по **iterator**;
current_member — текущий элемент агрегата (при его наличии), на который указывает **iterator**. Объект **iterator** не указывает текущий элемент, если он установлен в начале агрегата, в конце агрегата или является элементом массива, значение которого не установлено.

9.4.2 Объект **entity_instance**

Объект **entity_instance** является экземпляром, основанным на **entity_definition** (см. 6.4.12). Данный тип объекта является супертипом всех типов объекта, экземпляры которых могут запрашиваться или обрабатываться посредством реализации СИДД. Атрибуты **entity_instance** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации. Отношение между **entity_instance** и его подтипами должно обеспечиваться всеми реализациями СИДД.

EXPRESS-спецификация
*)
```
ENTITY entity_instance
    ABSTRACT SUPERTYPE OF (ONEOF(sdai_instance, application_instance));
END_ENTITY;
(*
```

Пример 8 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимая для экземпляра объекта.

```
ENTITY entity_instance
ABSTRACT SUPERTYPE OF (ONEOF(sdai_instance, application_instance));
    owning_model  : sdai_model;
    definition    : entity_definition;
    values        : LIST [0:?] OF attribute_value;
WHERE
    WR1 : SELF IN SELF\owning_model.contents.instances;
END_ENTITY;
```

Определения атрибутов
owning_model — СИДД-модель (**sdai_model**), к которой принадлежит данный **entity_instance**;
definition — тип объекта, которым является данный экземпляр;
values — значения явных атрибутов данного экземпляра объекта (**entity_instance**).

9.4.3 Объект **application_instance**

Объектом **application_instance** является объект **entity_instance**, определение которого есть тип объекта, установленный в прикладной схеме или импортированный в нее. Атрибуты **application_instance** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации. Отношение подтипа между **application_instance** и **entity_instance** должно обеспечиваться во всех реализациях СИДД.

EXPRESS-спецификация
*)
```
ENTITY application_instance
    SUBTYPE OF (entity_instance);
END_ENTITY;
(*
```

Пример 9 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимая для прикладного экземпляра.

```
ENTITY application_instance
    SUBTYPE OF (entity_instance);
    persistent_label : OPTIONAL STRING;
END_ENTITY;
```

Определение атрибута
persistent_label — имя (при его наличии), связанное с прикладным экземпляром (см. 10.11.6).

9.4.4 Объект **sdai_instance**

Объект **sdai_instance** является объектом **entity_instance**, определение которого есть тип объекта, установленный в схеме СИДД. Отношения подтипа между **sdai_instance** и **entity_instance** и **dictionary_instance**, **session_instance** и **sdai_instance** должны обеспечиваться всеми реализациями СИДД.

EXPRESS-спецификация

```
*)
ENTITY sdai_instance
    ABSTRACT SUPERTYPE OF (ONEOF(dictionary_instance, session_instance))
    SUBTYPE OF (entity_instance);
END_ENTITY;
(*
```

9.4.5 Объект **dictionary_instance**

Объект **dictionary_instance** является объектом **sdai_instance**, определение которого есть тип объекта, установленный в схеме словаря СИДД (см. раздел 6). Отношение подтипа между **dictionary_instance** и **sdai_instance** должно обеспечиваться всеми реализациями СИДД.

EXPRESS-спецификация

```
*)
ENTITY dictionary_instance
    SUBTYPE OF (sdai_instance);
END_ENTITY;
(*
```

9.4.6 Объект **session_instance**

Объект **session_instance** является объектом **sdai_instance**, определение которого есть тип объекта, установленный в схеме сеанса СИДД или импортированный в данную схему из схемы совокупности СИДД (см. разделы 7 и 8). Типы объектов, импортированные в схему сеанса СИДД из схемы словаря СИДД, не являются экземплярами **session_instance.** Отношение подтипа между **session_instance** и **sdai_instance** должно обеспечиваться всеми реализациями СИДД.

EXPRESS-спецификация

```
*)
ENTITY session_instance
    SUBTYPE OF (sdai_instance);
END_ENTITY;
(*
```

9.4.7 Объект **attribute_value**

Объект **attribute_value** является значением, связанным с атрибутом **entity_instance**. Атрибуты объекта **attribute_value** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

```
*)
ENTITY attribute_value;
END_ENTITY;
(*
```

Пример 10 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимая для значения атрибута.

```
ENTITY attribute_value;
    data_value              : OPTIONAL primitive;
    attribute_definition    : attribute;
DERIVE
    value set: BOOLEAN  := EXISTS(data_value);
END_ENTITY;
```

Определения атрибутов

data_value — значение (при его наличии), связанное с атрибутом; если оно отсутствует — атрибут не представлен;

attribute_definition — определение атрибута из схемы словаря, для которого оно имеет значение;

value_set — булевская величина, соответствующая TRUE, если атрибут имеет значение, FALSE, если значение атрибута не установлено.

9.4.8 Объект **select_value**

Объект **select_value** является значением, связанным с типом данных, определяемым конструкцией TYPE языка EXPRESS, устанавливающей тип SELECT. Объект **select_value** содержит информацию, достаточную для однозначной идентификации возможного пути, определяющего значение типа в случае, когда схема устанавливает граф типов, возможно включающий тип SELECT, а без дополнительной информации относительно типа такое определение может быть неоднозначным. Атрибуты **select_value** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

```
*)
ENTITY select_value;
    SUPERTYPE OF (ONEOF(select_aggregate_instance));
END_ENTITY;
(*
```

Пример 11 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимая для значения выбора.

```
ENTITY select_value;
    SUPERTYPE OF (ONEOF(select_aggregate_instance));
    data_value  : OPTIONAL primitive;
    data_type   : LIST [0:?] OF defined_type;
END_ENTITY;
```

Определения атрибутов

data_value — значение, связанное с типом данных;

data_type — типы, достаточные для идентификации конкретных значений в порядке их представления прикладным программистом.

9.4.9 Объект **select_aggregate_instance**

Объект **select_aggregate_instance** является объектом **select_value**, имеющим в качестве своего значения **data_value** экземпляра агрегата.

EXPRESS-спецификация

```
*)
ENTITY select_aggregate_instance;
    SUBTYPE OF (select_value);
END_ENTITY;
(*
```

Неформальное утверждение

IP1 — data_value должно быть экземпляром агрегата.

9.4.10 Объект **enumeration_value**

Объект **enumeration_value** является значением, связанным с типом данных, являющихся типом ENUMERATION языка EXPRESS. Атрибуты **enumeration_value** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

```
*)
ENTITY enumeration_value;
END_ENTITY;
(*
```

Пример 12 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимых для перечисляемого значения.

```
ENTITY enumeration_value;
    enumeration_name      : INTEGER;
    enumeration_typename : defined_tupe;
END_ENTITY;
```

Определения атрибутов

enumeration_name — позиция внутри списка **enumeration_type.elements**, соответствующая элементу перечисления, значение которого связано с типом данных (см. 6.4.28);

enumeration_typename — объект **defined_type**, определяющей элемент перечисления, который является значением типа данных.

Неформальное утверждение

IP1 — объект **defined_type**, на который дана ссылка в **enumeration_typename**, должен быть разрешен посредством конструкции TYPE в схеме так, чтобы он мог иметь в качестве значения атрибута **defined_type.domain** тип **enumeration_type.**

9.4.11 Объект **aggregate_instance**

Объект **aggregate_instance** является экземпляром **aggregate_type**.

EXPRESS-спецификация

```
*)
ENTITY aggregate_instance
    ABSTRACT SUPERTYPE OF (ONEOF(unordered_collection, ordered_collection));
END_ENTITY;
(*
```

9.4.12 Объект **unordered_collection**

Объект **unordered_collection** является объектом **aggregate_instance**, представляющим собой экземпляр типа мультимножества или набора (bag или set).

EXPRESS-спецификация

```
*)
ENTITY unordered_collection
    ABSTRACT SUPERTYPE OF (ONEOF(set_instance, bag_instance))
    SUBTYPE OF (aggregate_instance);
END_ENTITY;
(*
```

9.4.13 Объект **set_instance**

Объект **set_instance** является объектом **unordered_collection**, представляющим собой экземпляр типа набора (set). Атрибуты **set_instance** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

```
*)
ENTITY set_instance
    SUBTYPE OF (unordered_collection);
END_ENTITY;
(*
```

Пример 13 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимых для экземпляра набора.

```
ENTITY set_instance
    SUBTYPE OF (unordered_collection);
    set_definition    : set_type;
    contents          : SET [0:?]OF primitive
END_ENTITY;
```

Определения атрибутов

set_definition — схема словаря, определяющая набор;

contents — значения в наборе.

9.4.14 Объект **bag_instance**

Объект **bag_instance** является объектом **unordered_collection**, представляющим собой экземпляр мультимножества. Атрибуты **bag_instance** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

```
*)
ENTITY bag_instance
    SUBTYPE OF (unordered_collection);
END_ENTITY;
(*
```

Пример 14 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимых для экземпляра мультимножества.

```
ENTITY bag_instance
    SUBTYPE OF (unordered_collection);
```

```
        bag_definition   : bag_type;
        contents         : BAG [0:?]OF primitive;
END_ENTITY;
```

Определения атрибутов

bag_definition — схема словаря, определяющая мультимножество;

contents — значения в мультимножестве.

9.4.15 Объект **ordered_collection**

Объект **ordered_collection** является объектом **aggregate_instance**, представляющим собой экземпляр типа списка или массива.

EXPRESS-спецификация

```
*)
ENTITY ordered_collection
        ABSTRACT SUPERTYPE OF (ONEOF(list_instance, array_instance))
        SUBTYPE OF (aggregate_instance);
END_ENTITY;
(*
```

9.4.16 Объект **list_instance**

Объект **list_instance** является объектом **ordered_collection**, представляющим собой экземпляр типа списка.

EXPRESS-спецификация

```
*)
ENTITY list_instance
        ABSTRACT SUPERTYPE OF (ONEOF(non_persistent_list_instance,
        schema_defined_list_instance))
        SUBTYPE OF (ordered_collection);
END_ENTITY;
(*
```

9.4.17 Объект **schema_defined_list_instance**

Объект **schema_defined_list_instance** является объектом **list_instance**, представляющим собой экземпляр типа списка, определенный в схеме СИДД или прикладной схеме. Атрибуты **schema_defined_list_instance** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

```
*)
ENTITY schema_defined_list_instance
        SUBTYPE OF (list_instance);
END_ENTITY;
(*
```

Пример 15 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимых для экземпляра списка, определенного в схеме.

```
ENTITY schema_defined_list_instance
        SUBTYPE OF (list_instance);
        list_definition   : list_type;
        contents          : LIST [0:?]OF primitive;
END_ENTITY;
```

Определения атрибутов

list_definition — схема словаря, определяющая список;

contents — значения в списке.

9.4.18 Объект **non_persistent_list_instance**

Объект **non_persistent_list_instance** является объектом **list_instance**, представляющим собой экземпляр непостоянного, неограниченного списка экземпляров объектов (**entity_instance**).

EXPRESS-спецификация

```
*)
ENTITY non_persistent_list_instance
        SUBTYPE OF (list_instance);
```

END_ENTITY;
(*

Пример 16 — Ниже описана примерная EXPRESS-спецификация функциональных возможностей, необходимых для экземпляра непостоянного списка.

ENTITY non_persistent_list_instance
    SUBTYPE OF (list_instance);
    contents : LIST [0:?]OF entity_instance;
END_ENTITY;

Определения атрибута

contents — значения в списке.

9.4.19 Объект **array_instance**

Объект **array_instance** является объектом **ordered_collection**, представляющим собой экземпляр типа массива. Допустимые значения индекса **array_instance** устанавливаются во время создания этого экземпляра. Последующие операции над массивом могут переопределять значения индекса для **array_instance**, зависящие от совокупности схемы, в которой определен данный тип массива (см. 10.18.3). Атрибуты **array_instance** описаны в настоящем стандарте, но соответствующая EXPRESS-спецификация отсутствует, так как нет необходимости в их реализации.

EXPRESS-спецификация

*)
ENTITY array_instance
    SUPERTYPE OF (ONEOF(application_indexed_array_instance))
    SUBTYPE OF (ordered_collection);
END_ENTITY;
(*

Определения атрибутов

array_definition — схема словаря, определяющая массив;

contents — значения примитивов, являющихся элементами массива. Индексы массива определяются как **bound_instance_value**.

9.4.20 Объект **application_indexed_array_instance**

Объект **application_indexed_array_instance** является объектом **array_instance**, представляющим собой экземпляр типа массива, верхний и нижний индексы которого устанавливаются приложением при создании данного экземпляра. Команда сброса индексов массива может возвращать в исходное состояние правильные позиции индексов для **application_indexed_array_instance** (см. 10.18.4).

EXPRESS-спецификация

*)
ENTITY application_indexed_array_instance
    SUBTYPE OF (array_instance);
END_ENTITY;
(*

*)
END_SCHEMA; -- SDAI_parameter_data_schema;
(*

10 Команды СИДД

10.1 Введение

В данном разделе определены команды СИДД, не определены порядок выполнения данных команд, входные и выходные данные для них и потенциальные указатели ошибок, обрабатываемые языковыми привязками СИДД.

Примечание 1 — Конкретная команда может отображаться одной или несколькими функциями в некоторых языковых привязках, могущими иметь разное количество передаваемые неявно входных и выходных данных или параметров ошибки, например имя функции в предыдущей привязке.

Описание каждой команды должно охватывать следующие аспекты:

- описание выполняемой функции или оказываемой услуги;

- входные данные: информацию, которую необходимо определить до выполнения команды (если она требуется). Каждый входной параметр определяется: именем, в котором каждое составляющее его слово начинается с заглавной буквы; типом параметра, соответствующим схемам из разделов 6—9, выделенными полужирным шрифтом; словами, набранными строчными буквами, определяющими тип имени, и текстовым описанием;

- выходные данные: информацию, доступную приложению после правильного выполнения команды (если она необходима). Выходные параметры определяются по тем же правилам, что и входные параметры.

Примечание 2 — Некоторые параметры одновременно могут быть и входными и выходными;

- возможные указатели ошибок: условия, приводящие к неправильному завершению команды. Текст, описывающий указатель ошибки, может отличаться от приведенного в разделе 11 более конкретным описанием типа параметра, связанного с указателем ошибки для данной команды;

- влияние среды СИДД: описание изменений входных параметров и экземпляров объектов схем из разделов 6—9 (при необходимости). Во многих случаях на атрибут определенного экземпляра объекта ссылаются посредством синтаксической конструкции **Parameter.attribute**, где **Parameter** является именем входного или выходного параметра, а **attribute** — атрибутом экземпляра объекта для **Parameter**.

10.2 Фундаментальные принципы и допущения

Применяют следующие фундаментальные принципы и допущения:

- характеристики параметров команд СИДД, описанные в схемах словаря, сеанса и совокупности СИДД. Отношения подтипов между типами экземпляров объекта, описанные в схеме параметрических данных СИДД;

- команды экземпляра объекта, применяемые ко всем экземплярам любого типа данных объекта, определенные в любой схеме СИДД или прикладной схеме, доступной для приложения;

- команды прикладного экземпляра, применяемые только к экземплярам типов данных объекта, определенным в прикладных схемах, доступных для приложения;

- когда требуется команда доступа к экземпляру объекта в СИДД-модели, неактивной в текущем сеансе, а хранилище, содержащее данную модель, открыто, к модели автоматически применяется функция, эквивалентная команде начала доступа в режиме «только чтение». Если же хранилище, содержащее СИДД-модель, закрыто, оно само и СИДД-модель не должны автоматически открываться, и должна выдаваться ошибка;

- когда команда завершается неправильно, значения входных параметров не должны изменяться. Воздействие на выходные параметры в настоящем стандарте не определено;

- для реализаций, не поддерживающих доступ к словарю данных СИДД, выполнение команд, определяющих экземпляр из схемы словаря, обеспечивается командами языковых привязок СИДД, в которых данные параметры определяются по имени;

- для команд, в которых атрибут экземпляра объекта является входным параметром, этот атрибут может быть явно определен в **entity_definition**, на котором базируется **entity_instance**, или унаследован из супертипа данного **entity_definition**. Для языковых привязок СИДД, дополнительно поддерживающих определенные параметры по именам, случай наследования атрибутов с теми же именами из различных супертипов должен обрабатываться следующим образом: перед именем атрибута ставится имя супертипа типа объекта, из которого данный атрибут унаследован (см. 9.2.3.3 ГОСТ Р ИСО 10303-11);

- поведение команд доступа к экземпляру агрегата через его идентификатор или итератор во время использования других команд для добавления, изменения, перемещения или удаления элементов данного экземпляра в настоящем стандарте не определено;

- все итераторы, чьи агрегаты были удалены, должны быть изъяты в конце сеанса СИДД или раньше, в зависимости от реализации или условий, диктуемых языком программирования соответствующей привязки. При этом команда СИДД, использующая итератор в качестве параметра, должна возвращать ошибку AI_NEXS или IR_NEXS;

- экземпляры набора, мультимножества и списка управляются командами СИДД относительно их границ иначе, чем экземпляры массива. Нижние и верхние границы экземпляров наборов, мультимножеств и списков рассматриваются как ограничения количества элементов, которое могут содержать правильные экземпляры этих объектов. Эти границы не влияют на команды создания, добавления или перемещения элементов из наборов, мультимножеств и списков;

- экземпляры массива управляются командами СИДД относительно их границ другим образом, чем экземпляры набора, мультимножества и списка. Нижний и верхний индексы для экземпляров массива рассматриваются как устанавливаемый размер и допустимые позиции индекса для правильного экземпляра массива. Размер и допустимые позиция индекса для экземпляров массива устанавливаются в момент создания экземпляра массива, и изменить их можно только явным вызовом команды переиндексирования. Так как язык EXPRESS допускает объявление индекса массива в зависимости от совокупности прикладной схемы, команды СИДД позволяют создавать экземпляры одного и того же типа массива, имеющие различные допустимые позиции индекса. В случаях, когда совокупность не позволяет однозначно вычислить выражение индекса, команды СИДД позволяют определить значение индекса посредством приложения.

10.3 Команды среды

10.3.1 Открытие сеанса

Данная команда инициализирует реализацию и начинает новый сеанс СИДД. Она также открывает хранилище, содержащее экземпляр и данные схемы сеанса, а закрыть это хранилище может только команда закрытия сеанса. Возможен непосредственный доступ к данным сеанса. В реализации, обеспечивающей доступ к словарю данных, хранилища и СИДД-модели, содержащие информацию словаря данных, этой командой не открываются и не доступны, пока не будет начата транзакция.

Выход

Сеанс: **sdai_session;**
экземпляр **sdai_session**, созданный командой.

Указатели возможных ошибок

SS_OPN Сеанс СИДД уже открыт.
SS_NAVL Реализация не может открыть сеанс.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Session (сеанс), создан правильный экземпляр **sdai_session**.

Создается правильный экземпляр реализации (**inplementation**), а его идентификатор устанавливается как значение **Session.sdai_implementation**.

Атрибут **Session.known_servers** должен быть инициализирован экземпляром набора, элементы которого являются экземплярами **sdai_repository**, доступными для приложения в данном сеансе. В данном экземпляре набора находится инициализирующий элемент **active_servers**.

Атрибут **active_servers** данного сеанса должен быть инициализирован экземпляром набора, состоящим из одного элемента — идентификатора хранилища, содержащего экземпляр схемы, основанной на схеме сеанса СИДД, и СИДД-модель, содержащую данные сеанса.

Атрибут **active_models** данного сеанса должен быть инициализирован экземпляром набора, состоящим из одного элемента — идентификатора СИДД-модели, в которой содержатся данные сеанса.

Атрибут **errors** данного сеанса должен быть инициализирован пустым экземпляром списка. Атрибута **recording_active** должен быть инициализирован со значением TRUE, если реализация обеспечивает описание событий, и FALSE — в противном случае.

Атрибут **data-dictionary** данного сеанса должен быть инициализирован экземпляром **schema_instance**, в котором связываются СИДД-модели, содержащие словарь данных, если реализация обеспечивает доступ к этим словарям.

10.4 Команды сеанса

10.4.1 Запись ошибки

Эта операция добавляет событие ошибки к записи ошибок сеанса СИДД.

Вход

Сеанс: **sdai_session**;
сеанс СИДД, в котором возникла ошибка.

Идентификатор функции: **string_value;**
строка, определяющая команду, с которой связана ошибка.

Ошибка: **integer_value**;
код ошибки для события ошибки.

Описание: **string_value**;
описание события ошибки.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
ER_NSET Запись событий не установлена
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Создан и добавлен к списку **error_event** новый экземпляр **error_event**, представляющий атрибут **Session.errors**.

10.4.2 Начало описания события

Данная команда разрешает или подтверждает доступ к описанию события для команд СИДД во время сеанса. Любые события ошибок, ранее описанные в течение сеанса, остаются в записи событий ошибок, а любые новые описанные события ошибок добавляются к данной записи.

Вход

Сеанс: **sdai_session**;
сеанс, в котором допускается описание.

Выход

Результат: **boolean_value**;
реализация СИДД возвращает значение TRUE, если описание событий обеспечивается и доступно, FALSE — если не обеспечивается.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Если значение атрибута **Session.sdai_implementation.recording_level** равно двум, что определяет обеспечение реализацией СИДД описания событий, то атрибуту **Session.recording_active** должно быть присвоено или оставлено значение TRUE.

10.4.3 Окончание описания события

Данная команда отключает описание событий для сеанса СИДД.

Вход

Сеанс: **sdai_session**;
сеанс, для которого прекращается описание событий.

Выход

Результат: **boolean_value**;
реализация СИДД возвращает значение TRUE, если описание событий обеспечивается и прекращено, FALSE — если не обеспечивается.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Атрибуту **Session.recording_active** должно быть присвоено или оставлено значение FALSE.

10.4.4 Закрытие сеанса

Данная команда прекращает сеанс СИДД. Дальнейшие команды СИДД возможны только после выполнения команды открытия сеанса. В реализациях, обеспечивающих уровни транзакции 1 или 2 (см. 13.1.1), реализация должна вести себя так, как будто выполнена команда закрытия хранилища для всех хранилищ из **session.active_servers**. В реализации, обеспечивающей уровень транзакции 3, реализация должна вести себя так, как будто выполнена команда окончания доступа и аварийного завершения транзакции, если транзакция существует в сеансе, а затем команда закрытия хранилища для каждого открытого хранилища независимо от того, существует транзакция или нет.

Вход

Сеанс: **sdai_session**;
закрываемый сеанс.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

В реализации, поддерживающей уровень транзакции 1 или 2, для всех **sdai_repository** в **Session.active_servers** должна быть выполнена функция, эквивалентная команде закрытия хранилища.

В реализации, поддерживающей уровень транзакции 3, для **Session.active_transaction** должна быть выполнена функция, эквивалентная команде окончания доступа и аварийного завершения транзакции. Для каждого **sdai_repository** в **Session.active_servers** должна быть выполнена функция, эквивалентная команде закрытия хранилища.

Все экземпляры всех типов объектов во всех схемах СИДД и прикладных схемах больше недоступны.

10.4.5 Открытие хранилища

Данная команда открывает содержимое хранилища для последующего доступа к нему.

Вход

Сеанс: **sdai_session;**
сеанс, в котором хранилище должно быть открыто.

Хранилище: **sdai_repository;**
открываемое хранилище.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NEXS Хранилище не существует.
RP_NAVL Хранилище недоступно в данном сеансе.
RP_OPN Хранилище уже открыто.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Хранилище (**repository**) должно быть добавлено к набору в **Session.active_servers**.

10.4.6 Начало транзакции с доступом «чтение—запись»

Данная команда определяет начало последовательности команд в сеансе, обеспечивающей доступ к экземплярам объектов с разрешением внесения изменений в данные экземпляры.

Вход

Сеанс: **sdai_session;**
сеанс, для которого открывается доступ в режиме «чтение—запись».

Выход

Транзакция: **sdai_transaction;**
открытая транзакция в режиме «чтение—запись».

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_EXS Транзакция уже открыта.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Создается допустимый экземпляр **sdai_transaction**.

Атрибут **Transaction.mode** должен быть установлен для **read_write**.

Данная транзакция (**transaction**) должна быть установлена как **Session.active_transaction**.

10.4.7 Начало транзакции с доступом «только чтение»

Данная команда определяет начало последовательности команд в сеансе, обеспечивающей доступ к экземплярам объектов без внесения изменений в данные экземпляры.

Вход

Сеанс: **sdai_session;**
сеанс, для которого открывается доступ в режиме «только чтение».

Выход

Транзакция: **sdai_transaction;**
открытая транзакция в режиме «только чтение».

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_EXS Транзакция уже открыта.
FN_NAVL Функция не обеспечивается данной реализацией.

SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Создается допустимый экземпляр **sdai_transaction**.

Атрибут **Transaction.mode** должен быть установлен для **read_only**.

Данная транзакция (**transaction**) должна быть установлена как **Session.active_transaction**.

10.4.8 Фиксация транзакции

Данная команда фиксирует все изменения содержимого транзакции, СИДД-моделей и экземпляров схем всех открытых хранилищ, внесенные с момента активизации последней команды: начала транзакции с доступом «чтение—запись», фиксации или прерывания транзакции, независимо от того, какая из команд была последней. При этом существующая транзакция в режиме «чтение—запись» остается активной. Данная команда не выполняет никаких действий в случае, если текущая транзакция находится в режиме «только чтение». Данная команда обновляет или устанавливает значение атрибута **change_date** для любого экземпляра схемы или СИДД-модели, которые были изменены или созданы.

Вход

Транзакция: **sdai_transaction;**
фиксируемая транзакция.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_EAB Транзакция прервана аварийно.
TR_NAVL Транзакция недоступна в текущем сеансе.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

В открытом хранилище делаются постоянными текущие состояния всех экземпляров следующих объектов:

- **sdai_repository_contents;**
- **schema_instance;**
- **sdai_model.name;**
- **sdai_model_contents;**
- **entity_extent;**
- **scope;**
- **application_instance;**
- **aggregate_instance,** за исключением **non_persistent_list_instance.**

Значение атрибута **schema_instance.change_date** любого измененного или созданного экземпляра схемы должно быть установлено для текущей даты.

Значение атрибута **sdai_model.change_date** для любой измененной или созданной СИДД-модели должно быть установлено для текущей даты.

10.4.9 Аварийное прерывание (abort)

Данная команда восстанавливает состояние содержимого транзакции, СИДД-моделей и экземпляров схем всех открытых хранилищ, существовавшее на момент активизации последней команды: начала транзакции с доступом в режиме «чтение—запись» или фиксации, независимо от того, какая команда была последней. Восстанавливаются все удаленные экземпляры объектов, все созданные экземпляры прекращают существование, а все изменения экземпляров уничтожаются. При этом существующая транзакция в режиме «чтение—запись» остается активной. Никаких действий по этой команде не выполняется, если текущая транзакция находится в режиме «только чтение».

Вход

Транзакция: **sdai_transaction;**
прерываемая транзакция.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_EAB Транзакция прервана аварийно.
TR_NAVL Транзакция недоступна в текущем сеансе.

FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Поведение итератора в экземпляре агрегата, созданном, измененном или удаленном с момента активизации последней команды начала транзакции с доступом в режиме «чтение—запись» или фиксация, в настоящем стандарте не определено.

Для следующих экземпляров в открытых хранилищах, созданных с момента активизации последней команды начала транзакции с доступом в режиме «чтение—запись» или фиксация, должна быть вызвана соответствующая команда удаления:

- **schema_instance;**
- **sdai_model;**
- **scope;**
- **application_instance;**
- **aggregate_instance** за исключением **non_persistent_list_instance.**

Для следующих экземпляров в открытых хранилищах, удаленных явно или неявно с момента последней активизации команды начала транзакции с доступом в режиме чтение—запись» или фиксация, должны быть восстановлены состояния, существовавшие до момента активизации этих команд:

- **schema_instance;**
- **sdai_model;**
- **sdai_model_contents;**
- **entity_extent;**
- **scope;**
- **application_instance;**
- **aggregate_instance** за исключением **non_persistent_list_instance.**

Состояние всех следующих экземпляров в открытых хранилищах, измененных с момента последней активизации команды начала транзакции с доступом в режиме «чтение—запись» или фиксация, должно быть возвращено в исходное положение, существовавшее до момента активизации этих команд:

- **sdai_repository_contents;**
- **schema_instance;**
- **sdai_model;**
- **sdai_model_contents;**
- **entity_extent;**
- **scope;**
- **application_instance;**
- **aggregate_instance** за исключением **non_persistent_list_instance.**

10.4.10 Завершение доступа и фиксации транзакции

Данная команда заканчивает последовательность команд, открытую командой начала транзакции с доступом в режиме «чтение—запись» или «только чтение». Реализация должна вести себя так, как будто была выполнена команда фиксации транзакции до завершения доступа к ней. Последующие команды доступа к экземплярам объекта в сеансе возможны только после вызова команд начала транзакции с доступом в режимах «чтение—запись» или «только чтение».

Вход

Транзакция: **sdai_transaction;**
прерываемая транзакция.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_EAB Транзакция прервана аварийно.
TR_NAVL Транзакция недоступна в текущем сеансе.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Транзакция (**transaction**) должна быть удалена.

10.4.11 Завершение доступа к транзакции и аварийное прерывание

Данная команда заканчивает последовательность команд, открытую командой начала транзакции с доступом в режиме «чтение—запись» или «только чтение». Реализация должна вести себя так, как будто была выполнена команда аварийного прерывания перед завершением доступа к транзакции. Последующие команды доступа к экземплярам объекта в сеансе возможны только после вызова команд начала транзакции с доступом в режимах «чтение—запись» или «только чтение».

Вход

Транзакция: **sdai_transaction;**
прерываемая транзакция.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_EAB Транзакция прервана аварийно.
TR_NAVL Транзакция недоступна в текущем сеансе.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Транзакция (**transaction**) должна быть удалена.

10.4.12 Создание нефиксированного списка

Данная команда создает пустой экземпляр **non_persistent_list_instance**. Нефиксированные списки обеспечивают услуги, такие как контейнер для результатов запроса и стандартные механизмы приложений СИДД для обработки коллекций экземпляров объекта. Нефиксированные списки необходимы только для обеспечения списков экземпляров объектов для типов объектов, найденных в словаре СИДД или в библиотеке функций, сгенерированных реализацией предшествующей языковой привязки. Невозможно присвоить идентификатор нефиксированного списка атрибуту экземпляра объекта. Все нефиксированные списки должны быть удалены в конце сеанса СИДД, и могут быть удалены раньше, в зависимости от предписаний языка программирования конкретной привязки. К нефиксированным спискам применяют следующие команды:

- 10.12.1—10.12.7;
- 10.13.2 и 10.13.3;
- 10.15.1—10.15.3;
- 10.16.1;
- 10.19.1—10.19.3 и 10.19.7.

Выход

Список: **non_persistent_list_instance;**
созданный нефиксированный список.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Создается пустой нефиксированный список (**non_persistent_list_instance**).

10.4.13 Удаление нефиксированного списка

Данная команда удаляет неограниченный нефиксированный список и не влияет на элементы данного списка.

Вход

Список: **non_persistent_list_instance;**
удаляемый нефиксированный список.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
AI_NEXS Нефиксированный список не существует.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Список (**list**) удаляется.

10.4.14 Запрос СИДД

Данная команда наполняет существующий нефиксированный список, реализуя запрос через источник. Источником может быть определен экземпляр агрегата, СИДД-модель, экземпляр схемы или хранилище. В случае, когда источником являются СИДД-модели, экземпляр схемы или хранилище, источник определяет набор СИДД-моделей и для каждой СИДД-модели из набора запрос выполняется через **entity_extent** типа объекта, определенного в логическом выражении запроса. После запроса результирующий агрегат будет содержать идентификаторы экземпляров тех объектов, для которых вычисленное логическое выражение имеет значение TRUE. Если результирующий список уже содержал идентификаторы каких-либо экземпляров объекта, они остаются в этом списке, а новые результаты запроса добавляют в конец этого списка. В случае, когда синтаксис логического выражения неверен, должны выдаваться ошибки VA_NVLD, OP_NVLD или AT_NVLD, определяющие соответствующие ошибки в значении, операторе или атрибуте. Если комбинации атрибута, значения и оператора не обеспечиваются, должна выдаваться ошибка VT_NVLD.

Логическое выражение, определяющее критерий, по которому формируется результат, должно быть строкой вида **VALUE OPERATOR entity{.attr_spec}.attr_spec,** где:

- **attr_spec** является определением имени атрибута в виде **attribute.name** или **entity_definition.name**, соединенным с **attribute.name** с разделением через точку (.) в случае, когда атрибуты с тем же именем унаследованы из многих супертипов;
- **{.attr_spec}** является пустым или содержащим несколько ссылок на объекты, значениям которых присвоены имена соответствующих атрибутов;
- **OPERATOR** — это =, <=, >=, <, >, <>, :=:, :<>:, IN, или LIKE;
- **VALUE** может быть ключевым словом UNSET или принимать значение, определенное в таблице 1, где UNSET проверяет наличие или отсутствие значения атрибута.

Простые виды могут комбинироваться с помощью NOT, AND, OR и круглых скобок для создания более сложных видов. Вычисление выражений в скобках определено в разделе 12 ГОСТ Р ИСО 10303-11. Логические операторы AND, OR и NOT определены соответственно в 12.4.2, 12.4.3 и 12.4.1 ГОСТ Р ИСО 10303-11.

Значения операторов сравнения =, <=, >=, <, > и <> определены в 12.2.1 ГОСТ Р ИСО 10303-11 с ограничением, что они не могут применяться к значениям типов данных агрегат или объект. Операторы сравнения экземпляров :=: и :<>: определены в 12.2.2 ГОСТ Р ИСО 10303-11 с ограничением, что они могут применяться только к значениям типов данных объекта.

Функция оператора LIKE, алгоритм сопоставления и символы сопоставления определены в 12.2.5 ГОСТ Р ИСО 10303-11, за исключением того, что левый операнд должен быть сопоставляемой строкой, а правый — контрольной строкой.

Функция оператора IN определена в 12.2.3 ГОСТ Р ИСО 10303-11.

В случае, когда VALUE в логическом выражении является литералом, формат этого литерала определен в 7.5 ГОСТ Р ИСО 10303-11.

Таблица 1 — Представления атрибутов, содержащихся в запросе

Представление атрибута	Значение оператора (OPERATOR)	Тип значения (VALUE)
simple_type; defined_type, областью значения которых является **simple_type**	=, <=, >=, <, >, <>	литерал в соответствующей области значений
вложенный **aggregation_type; defined_type,** область значения которых вычисляется для вложенного **aggregation_type**	**all**	не обеспечивается
entity instance; defined_type, область значения которых вычисляется для **entity_instance**	:=:, :<>:	лексема «ENTITY»
defined_type, область значения которого вычисляется для **enumeration_type**	=, <=, >=, <, >, <>	строковый литерал
string_type; defined_type, область значения которых вычисляется для **string_type**	**LIKE**	строковый литерал
aggregation_type; defined_type, область значения которых вычисляется для **aggregation_type**	**IN**	лексема «ENTITY» или литерал соответствующей области значений

Вход

Источник: **query_source;**
невложенный, фиксированный или нефиксированный агрегат, СИДД-модель, экземпляр схемы или хранилище.

Критерий: **string_value;**
логическое выражение, определяющее критерий, по которому устанавливается необходимость включения в результат.

Объект: **entity_instance;**
значение для ENTITY в случае, когда запрашиваемый атрибут является ссылкой на тип данных объекта.

Результат: **non_persistent_list_instance;**
нефиксированный агрегат, в который внесены идентификаторы экземпляров объектов, удовлетворяющих заданному критерию.

Выход

Количество: **integer_value;**
целочисленное количество числа экземпляров объекта, удовлетворивших заданному критерию.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
AI_NEXS Экземпляр входного или выходного агрегата не существует.
RP_NEXS Хранилище не существует.
MO_NEXS СИДД-модель не существует.
SI_NEXS Экземпляр схемы не существует.
EI_NEXS Экземпляр объекта не существует.
EI_NVLD Логическое выражение экземпляра объекта неверно.
VA_NVLD Значение логического выражения неверно.
OP_NVLD Логическое выражение оператора неверно.
AT_NVLD Логическое выражение атрибута неверно.
VT_NVLD Логическое выражение комбинации значения, оператора и атрибута неверно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Элементы источника (**source**), удовлетворившие заданному критерию, вносятся в качестве элементов в результат (**result**).

10.5 Команды хранилища

10.5.1 Создание СИДД-модели

Данная команда создает новую СИДД-модель, внутри которой могут создаваться и быть доступными экземпляры объектов. Вновь созданная модель не имеет связанного с ней режима доступа.

Вход

Хранилище: **sdai_repository;**
хранилище, в котором создается СИДД-модель.

Имя Модели: **string_value;**
имя новой СИДД-модели.

Схема: **schema_definition;**
схема, на которой должна быть основана данная СИДД-модель.

Выход

Модель: **sdai_model;**
вновь созданная СИДД-модель.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NEXS Хранилище не существует.
RP_NOPN Хранилище не открыто
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция не доступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.

MO_DUP Существует дубликат имени СИДД-модели.
VT_NVLD Тип значение имени СИДД-модели неверен.
SD_NDEF Определение схемы отсутствует.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Должны быть созданы новые экземпляры **sdai_model** и **sdai_model_contents.**

Для ИмениМодели (**ModelName**) должен быть установлен атрибут **Model.name.**

Для Схемы (**Schema**) должен быть установлен атрибут **Model.underlying_schema.**

Атрибут **Model.mode** не будет установлен.

Для Хранилища (**Repository**) должен быть установлен атрибут **Model.repository.**

Для текущей даты должен быть установлен атрибут **Model.change_date.**

Для идентификатора вновь созданного экземпляра **sdai_model_contents** должен быть установлен атрибут **Model.contents.**

Атрибуты **Model.contents.instances** и **Model.contents.populated_folders** будут инициализированы экземплярами набора, не содержащего элементов.

Атрибут **Model.contents.folders** должен содержать единственный экземпляр **entity_extent** для каждого **entity_definition** в **Schema.entities**, у которого атрибут **entity_extent.definition** установлен соответствующим **entity_definition**, а атрибут **entity_extent.instances** будет инициализирован экземпляром набора, не содержащего элементов.

Для включения Модели (**Model**) должен быть обновлен атрибут **Repository.contents.models.**

10.5.2 Создание экземпляра схемы

Данная команда устанавливает новый экземпляр схемы.

Вход

Хранилище: **sdai_repository;**
хранилище, в котором создается экземпляр схемы.

Имя: **string_value;**
имя нового экземпляра схемы.

Схема: **schema_definition;**
схема, на которой основан данный экземпляр схемы.

Выход

Экземпляр: **schema_instance;**
вновь созданный экземпляр схемы.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NEXS Хранилище не существует.
RP_NOPN Хранилище не открыто
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SI_DUP Существует дубликат имени экземпляра схемы.
VT_NVLD Тип значение имени экземпляра схемы неверен.
SD_NDEF Определение схемы отсутствует.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Экземпляр (**Instance**), создается допустимый **schema_instance.**

Для Имени (**Name**) должен быть установлен атрибут **Instance.name.**

Атрибут **Instance.associated_models** инициализирован для экземпляра набора без элементов.

Для Схемы (**Schema**) должен быть установлен атрибут **Instance.native_schema.**

Для Хранилища (**Repository**) должен быть установлен атрибут **Instance.repository.**

Для текущей даты должны быть установлены атрибуты **Instance.change_date** и **Instance.validation_date.**

Атрибуту **Instance.validation_result** должно быть установлено значение FALSE.

Атрибуту **Instance.validation_level** будет установлено то же значение, что и атрибуту **sdai_session.sdai_implementation.expression_level** для экземпляра **sdai_session**, где Хранилище (**Repository**) находится в **active_servers.**

Для включения Экземпляра (**Instance**) должен быть обновлен атрибут **Repository.contents.schemas.**

10.5.3 Закрытие хранилища

Данная команда закрывает ранее открытое хранилище. Экземпляры схем и СИДД-модели внутри этого хранилища больше не доступны для работы.

Реализация, обеспечивающая транзакции уровней 1 и 2 (см. 13.1.1), должна вести себя так, как будто команда завершения доступа в режиме «только чтение» применена для всех ранее открытых СИДД-моделей с доступом в данном режиме. Реализация, обеспечивающая транзакцию уровня 1, должна вести себя так, как будто команда завершения доступа в режиме «чтение—запись» ко всем СИДД-моделям с режимом «чтение—запись» применена для всех активных СИДД-моделей внутри хранилища. Реализация, обеспечивающая транзакцию уровня 2, должна вести себя так, как будто команды отмены изменений и завершения доступа в режиме «чтение—запись» ко всем СИДД-моделям с режимом «чтение—запись» применены для всех активных СИДД-моделей внутри хранилища.

В реализации, обеспечивающей транзакцию уровня 3, если транзакция с доступом «чтение—запись» активна в сеансе, имеют место два обстоятельства, при которых данная команда должна в результате привести к ошибке TR_RW. Во-первых, если какая-нибудь СИДД-модель внутри хранилища была создана, изменена или удалена после последнего применения команды фиксация или аварийное прерывание. Во-вторых, если какой-нибудь экземпляр схемы внутри хранилища был создан, изменен или удален после последнего применения команды фиксация или аварийное прерывание. В остальных случаях реализация, обеспечивающая транзакцию уровня 3, должна вести себя так, как будто команда завершения доступа в режиме «только чтение» применена ко всем активным СИДД-моделям, доступным в режиме «только чтение», а команда завершения доступа в режиме «чтение—запись» — ко всем СИДД-моделям, доступным в режиме «чтение—запись», внутри хранилища до его закрытия.

Вход

Хранилище: **sdai_repository;**
закрываемое хранилище.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NEXS Хранилище не существует.
RP_NOPN Хранилище не открыто.
TR_RW Транзакция типа «чтение—запись» и изменения запрещены.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

В текущем сеансе Хранилище (**Repository**) удаляется из **sdai_session.active_servers.**

В текущем сеансе все элементы **Repository.contents.models** удаляются из **sdai_session.active_models.**

10.6 Команды экземпляра схемы

10.6.1 Удаление экземпляра схемы

Данная команда удаляет экземпляр схемы. Если существуют ссылки между двумя СИДД-моделями, связанные с этим экземпляром схемы, и нет другого экземпляра схемы, с которым были бы связаны обе модели, тогда ссылки между экземплярами объектов в этих двух моделях являются неверными (см. 10.10.7).

Вход

Экземпляр: **schema_instance;**
удаляемый экземпляр схемы.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
SI_NEXS Экземпляр схемы не существует.
RP_NOPN Хранилище не открыто.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Экземпляр (**Instance**) удаляется.

Экземпляр (**Instance**) должен быть удален из **Instance.repository.contents.schemas.**

10.6.2 Переименование экземпляра схемы

Данная команда присваивает новое имя экземпляру схемы.

Вход

Экземпляр: **schema_instance;**
переименуемый экземпляр схемы.

Имя: **string_value;**
новое имя экземпляра схемы.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
SI_DUP Существует дубликат имени экземпляра схемы.
VT_NVLD Тип значение имени экземпляра схемы неверен.
SI_NEXS Экземпляр схемы не существует.
RP_NOPN Хранилище не открыто.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для Имени (**Name**) должен быть установлен атрибут **Instance.name.**

10.6.3 Добавление СИДД-модели

Данная команда добавляет СИДД-модель к набору СИДД-моделей, связанных с экземпляром схемы. Это позволяет экземплярам объектов в данной СИДД-модели ссылаться на экземпляры объектов в других моделях, связанных с данным экземпляром схемы, и одновременно быть ссылками для них. При этом также добавляются экземпляры объектов из данной СИДД-модели к области значений для проверки глобальных правил и правил уникальности, определенных экземпляром схемы. Если СИДД-модель не основана на той же схеме, что и данный экземпляр схемы, но базируется на внешней схеме, тогда экземпляр объекта в этой модели должен считаться связанным с данным экземпляром схемы, только если их типы объектов определены как эквивалентные по области значений с типом объекта из собственной схемы, на которой основан данный экземпляр схемы (см. А.2). Если эквивалентная области значений не обеспечивается, а добавляемая СИДД-модель основана на внешней схеме, должна выдаваться ошибка FN_NAVL.

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, с которым связана СИДД-модель.

Модель: **sdai_model;**
СИДД-модель, связанная с экземпляром схемы.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
SI_NEXS Экземпляр схемы не существует.
RP_NOPN Хранилище не открыто.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MO_NEXS СИДД-модель не существует.
MO_NDEQ СИДД-модель не эквивалентна по области значений с экземпляром схемы.
FN_NAVL Эквивалентность области значений не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Атрибут **Instance.associated_models** должен быть изменен для включения Модели (**Model**).

10.6.4 Удаление СИДД-модели

Данная команда удаляет СИДД-модель из набора СИДД-моделей, связанных с экземпляром схемы. Если данная СИДД-модель больше не имеет общего экземпляра схемы с другой СИДД-моделью в данном экземпляре схемы, все ссылки между двумя этими моделями станут не верными (см. 10.10.7).

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, из которого удаляется СИДД-модель.

Модель: **sdai_model;**
СИДД-модель, удаляемая из экземпляра схемы.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
SI_NEXS Экземпляр схемы не существует.
RP_NOPN Хранилище не открыто.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MO_NEXS СИДД-модель не существует.
MO_NVLD СИДД-модель не связана с экземпляром схемы.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Модель (**Model**) должна быть удалена из **Instance.associated_models.**

10.6.5 Проверка глобального правила

Данная команда определяет, удовлетворяет ли совокупность, связанная с экземпляром схемы, глобальному правилу, установленному в схеме. Экземплярами объектов, охватываемыми проверкой, являются все экземпляры типов объектов, к которым во всех СИДД-моделях, связанных с экземпляром схемы, применяется глобальное правило. Экземпляры объектов внутри СИДД-моделей, основанных на внешней схеме, включаются в проверку, если они являются экземплярами типов объектов, определенных экземпляром **external_schema** эквивалентными по области значений с типами объектов в собственной схеме. Включенные таким образом экземпляры объектов должны быть обработаны подобно экземплярам собственного типа, как определено в **domain_equivalent_type.** Если во внешнем типе сущности отсутствуют свойства, необходимые для удовлетворения правила, выдается ошибка ED_NVLD. Ссылки на экземпляры объектов в СИДД-моделях, которые не связаны с данным экземпляром схемы, должны обрабатываться так, как если бы они не были установлены.

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, ограничивающий проверку.

Правило: **global_rule;**
проверяемое глобальное правило.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) — FALSE, то это существующий нефиксированный список, в который добавляются те экземпляры **where_rule** внутри Правила (**Rule**), которым Экземпляр (**Instance**) не соответствует.

Выход

Результат: **logical_value;**
TRUE, если правило удовлетворено, FALSE, если правило не удовлетворено, и UNKNOWN, если вычисление выражения не определено или имеет значение UNKNOWN.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RU_NDEF Правило не определено.
SI_NEXS Экземпляр схемы не существует.
AI_NEXS Экземпляр агрегата не существует.
ED_NVLD Определение внешнего объекта неверно для правила.
RP_NOPN Хранилище не открыто.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Оценка выражения правила не обеспечивается данной реализацией.
EN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.6.6 Проверка правила уникальности (однозначности)

Эта операция определяет, удовлетворяет ли совокупность, связанная с экземпляром схемы, правилу уникальности (однозначности), установленному в схеме. Экземплярами объектов, охвачен-

ными проверкой, являются все экземпляры типов объектов, для которых объявлено данное правило, во всех СИДД-моделях, связанных с данным экземпляром схемы. Экземпляры объектов внутри СИДД-моделей, основанных на внешней схеме, включают в проверку, если они являются экземплярами типов объектов, определенных экземпляром **external_schema** эквивалентными по области значений с типами объектов в собственной схеме. Включенные таким образом экземпляры объектов должны быть обработаны как экземпляры собственного типа, что определено в **domain_equivalent_type**. Если во внешнем типе сущности отстутствуют свойства, необходимые для удовлетворения правила, выдается ошибка ED_NVLD. Ссылки на экземпляры объектов в СИДД-моделях, которые не связаны с данным экземпляром схемы, должны обрабатываться так, как если бы они не были установлены.

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, ограничивающий проверку.

Правило: **uniqueness_rule;**
проверяемое правило уникальности (однозначности).

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**)—FALSE, то это существующий нефиксированный список, в который добавляются экземпляры объектов, не прошедшие проверку.

Выход

Результат: **logical_value;**
TRUE, если правило удовлетворено, FALSE, если правило не удовлетворено, и UNKNOWN, если не установлены необязательные явные атрибуты, значение вычисляемого атрибута не определено или имеет значение UNKNOWN, или инверсный атрибут не имеет значения.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RU_NDEF Правило не определено.
SI_NEXS Экземпляр схемы не существует.
AI_NEXS Экземпляр агрегата не существует.
RP_NOPN Хранилище не открыто.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Оценка выражения правила не обеспечивается данной реализацией.
EN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.6.7 Проверка области значений ссылки на экземпляр

Данная команда определяет, все ли атрибуты в заданном прикладном экземпляре со ссылкой на экземпляр объекта имеют соответствующие значения, ссылающиеся на экземпляры объектов внутри СИДД-моделей, связанных с заданным экземпляром схемы.

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, ограничивающий проверку.

Объект: **application_instance;**
проверяемый прикладной экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**)—FALSE, то это существующий нефиксированный список, в который добавляются экземпляры типа атрибута, ссылающиеся на прикладные экземпляры, не связанные с данным Экземпляром (**Instance**).

Выход

Результат: **logical_value;**
TRUE, если все заданные атрибуты Объекта (**Object**) относятся к экземплярам объектов в Экземпляре (**Instance**), FALSE, если нет, и UNKNOWN, если значения требуемых явных атрибутов не обеспечивают ссылки из экземпляра объекта.

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
EI_NEXS	Экземпляр объекта не существует.
SI_NEXS	Экземпляр схемы не существует.
AI_NEXS	Экземпляр агрегата не существует.
RP_NOPN	Хранилище не открыто.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.6.8 Проверка экземпляра схемы

Данная команда определяет, удовлетворяет ли совокупность, связанная с экземпляром схемы, всем ограничениям, заданным внутри схемы, на которой основан данный экземпляр схемы. Данная команда обновляет информацию проверки, проводимой внутри данного экземпляра схемы.

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, ограничивающий проверку.

Выход

Результат: **logical_value;**
TRUE, если удовлетворены все ограничения схемы, на которой основан Экземпляр (**Instance**), FALSE, если нарушено какое-либо ограничение, и UNKNOWN, если результатом проверки какого-либо ограничения является значение UNKNOWN.

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
SI_NEXS	Экземпляр схемы не существует.
RP_NOPN	Хранилище не открыто.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
TR_NRW	Транзакция не имеет типа «чтение—запись».
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для текущей даты должен быть установлен атрибут **Instance.validation_date.**

Для объекта **implementation.expression_level** текущей реализации должен быть установлен атрибут **Instance.validation_level**.

Для объекта Результат (**Result**) должен быть установлен атрибут **Instance.validation_result**.

10.6.9 Определение актуальности проверки

Данная команда определяет, нужна ли полная проверка правильности экземпляра схемы после выполнения последней команды проверки экземпляра схемы, исходя из того, задано ли значение атрибута **schema_instance.validation_result** или внесено какое-либо изменение в экземпляры схемы или СИДД-модели, связанные с этим экземпляром схемы. При не установленном результате проверки или наличии любого изменения, данная команда определяет, что проверка не актуальна.

Вход

Экземпляр: **schema_instance;**
экземпляр схемы, ограничивающий проверку.

Выход

Результат: **boolean_value;**
TRUE, если текущий результат проверки имеет значение TRUE и с момента последней проверки не обнаружено изменений ни Экземпляра (**Instance**), ни элементов множества **Instance.associated_models,** в противном случае — FALSE.

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
SI_NEXS	Экземпляр схемы не существует.

RP_NOPN Хранилище не открыто.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.7 Команды СИДД-модели

10.7.1 Удаление СИДД-модели

Данная команда удаляет СИДД-модель (**sdai_model**) вместе со всеми содержащимися в ней экземплярами объектов (**entity_instances**), агрегатов (**aggregate_instances**) и областями действия (**scopes**). Любая последующая команда, использующая ссылку на данную модель или ее составную часть, должна вести себя так, как если бы эти ссылки отсутствовали.

Вход

Модель: **sdai_model;**
удаляемая модель.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Model и **Model.contents** должны быть удалены.

Каждый **entity_instance** из множества **Model.contents.instances** должны быть удалены так, как если бы к ним была применена команда удаления прикладного экземпляра (см. 10.11.2).

Должна быть удалена **Model** из множества **sdai_session.active_models** для текущего сеанса.

Должна быть удалена **Model** из множества **sdai_repository_contents.models** для хранилища, в котором была создана **Model**.

Должна быть удалена **Model** из множества **schema_instance.associated_models** для экземпляра схемы, с которыми была связана **Model**.

Должна быть удалена каждая **scope**, созданная внутри **Model**.

10.7.2 Переименование СИДД-модели

Данная команда присваивает новое имя **sdai_model**.

Вход

Модель: **sdai_model;**
переименуемая **sdai_model**.
ИмяМодели: **string_value;**
новое имя для **sdai_model.**

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
VT_NVLD Тип значения имени СИДД-модели неверен.
MO_DUP Существует дубликат имени СИДД-модели.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для объекта **ModelName** должен быть установлен атрибут **Model.name**.

10.7.3 Начало доступа «только чтение»

Данная команда делает доступными экземпляры внутри СИДД-модели, но ограничивает доступ к ним режимом «только чтение». В результате любой последующей команды СИДД, пытающейся изменить экземпляры внутри **sdai_model**, должна быть выдана ошибка.

Вход

Модель: **sdai_model;**

СИДД-модель, доступная в режиме «только чтение».

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
MX_RO СИДД-модель доступна в режиме «только чтение».
MX_RW СИДД-модель доступна в режиме «чтение—запись».
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для объекта **read_only** должен быть установлен атрибут **Model.mode**.

Объект **Model** должен быть добавлен к множеству **sdai_session.active_models** для текущего сеанса.

10.7.4 Перевод СИДД-модели в режим «чтение—запись»

Данная команда разрешает доступ в режиме «чтение—запись» к экземплярам СИДД-модели (**sdai_model**), к которой была применена команда начала доступа «только чтение» или которая была автоматически начата (открыта) с доступом «только чтение» в результате ссылки (обращения) к экземпляру объекта внутри этой модели.

Вход

Модель: **sdai_model;**

СИДД-модель, к которой разрешается доступ в режиме «чтение—запись».

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
MX_NDEF Доступ к СИДД-модели не определен.
MX_RW СИДД-модель доступна в режиме «чтение—запись».
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для объекта **read_write** должен быть установлен атрибут **Model.mode**.

10.7.5 Завершение доступа «только чтение»

Данная команда завершает доступ к СИДД-модели (**sdai_model**) в режиме «только чтение». Последующие команды над экземплярами, содержащимися внутри данной модели, не будут выполняться, пока не начнется доступ к этой модели при помощи команды начала доступа «только чтение» или «чтение—запись», или после автоматического начала доступа к этой модели в режиме «только чтение» в результате использования ссылки на экземпляр объекта внутри данной модели.

Вход

Модель: **sdai_model;**

модель, к которой прекращается доступ.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
MX_NDEF Доступ к СИДД-модели не определен.
MX_RW СИДД-модель доступна в режиме «чтение—запись».
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Атрибут **Model.mode** должен стать неопределенным.

Объект **Model** должен быть удален из множества **sdai_session.active_models** для текущего сеанса.

10.7.6 Начало доступа «чтение—запись»

Данная команда делает доступными экземпляры внутри СИДД-модели и разрешает доступ к ним в режиме «чтение—запись».

Вход

Модель: **sdai_model;**
СИДД_модель, к которой разрешается доступ в режиме «чтение—запись».

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NEXS Транзакция не открыта.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
MX_RO СИДД-модель доступна в режиме «только чтение».
MX_RW СИДД-модель доступна в режиме «чтение—запись».
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для объекта **read_write** должен быть установлен атрибут **Model.mode**.

Объект **Model** должен быть добавлен к множеству **sdai_session.active_models** для текущего сеанса.

10.7.7 Завершение доступа «чтение—запись»

Данная команда завершает доступ к СИДД-модели (**sdai_model**) в режиме «чтение—запись». Последующие команды над экземплярами, содержащимися внутри данной модели, не будут выполняться, пока не начнется доступ к этой модели при помощи команды начала доступа «только чтение» или «чтение—запись», или после автоматического начала доступа к этой модели в режиме «только чтение» в результате использования ссылки на экземпляр объекта внутри данной модели. Реализация, обеспечивающая транзакцию уровня 2, должна вести себя так, как будто к данной СИДД-модели была применена команда отмены изменений. В реализации, обеспечивающей транзакцию уровня 3, если какой-либо прикладной экземпляр (**application_instance**) или область действия (**scope**) внутри данной СИДД-модели были созданы, изменены или удалены после выполнения последней команды фиксации, прерывания или начала транзакции с доступом «чтение—запись», описываемая команда должна выдать ошибку TR_RW.

Вход

Модель: **sdai_model;**
модель, к которой прекращается доступ.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
MX_RO СИДД-модель доступна в режиме «только чтение».
MX_NDEF Доступ к СИДД-модели не определен.
TR_RW Транзакция имеет тип «чтение—запись» и изменения запрещены.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Атрибут **Model.mode** должен быть возвращен в неустановленное значение.

Объект **Model** должен быть удален из множества **sdai_session.active_models** для текущего сеанса.

10.7.8 Получение определения объекта

Данная команда возвращает идентификатор **entity_definition** из словаря данных, основываясь на имени объекта из схемы, на которой базируется заданная СИДД-модель.

Вход

Модель: **sdai_model;**
модель, основанная на схеме, в которой производится поиск.

ИмяОбъекта: **string_value;**
имя типа объекта, представляющего интерес.

Выход

Объект: **entity_definition;**
словарный экземпляр **entity_definition** из множества **Model.underlying_schema.entities**, имеющий значение атрибута **entity_definition.name = EntityName**.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
RP_NOPN Хранилище не открыто.
MO_NEXS СИДД-модель не существует.
ED_NDEF Имя определения объекта не установлено.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.7.9 Создание экземпляра объекта

Данная команда создает новый экземпляр объекта заданного типа данных объекта. Атрибуты этого экземпляра изначально не установлены, чтобы команда проверки (тестирования) атрибута возвращала значение FALSE. Данная команда применима только к экземплярам типов объектов, объявленным в прикладных схемах.

Вход

Тип: **entity_definition;**
тип объекта создаваемого экземпляра.

Модель: **sdai_model;**
модель, которая будет содержать экземпляр объекта.

Выход

Предмет: **application_instance;**
вновь созданный экземпляр объекта.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
TR_NRW Транзакция не имеет типа «чтение—запись».
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
ED_NDEF Определение объекта не установлено.
ED_NVLD СИДД-модель и определение объекта не базируются на одной и той же схеме.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Атрибут **Model.contents.instances** должен быть изменен включением в него Предмета (**Object**).

В **entity_extent** множества **Model.contents.folders**, для которого значение атрибута **entity_extent.definition = Type**, атрибут **entity_extent.instances** должен быть изменен включением в него **Object**. Тот же самый **entity_extent** должен быть внесен в множество **Model.contents.populated_folders**.

10.7.10 Отмена изменений

Данная команда восстанавливает состояние содержания, включая прикладные экземпляры (**application_instances**) и связанные с ними области действия (**scope**), СИДД-модели, существовавшее до последнего выполнения команды сохранения изменений или начала доступа «чтение—запись», в зависимости от того, какая из команд выполнялась последней. Восстанавливаются все удаленные экземпляры объектов, все созданные экземпляры удаляются, а все изменения экземпляров не сохраняются. Данная команда выполнима только для моделей, к которым разрешен текущий доступ в режиме «чтение—запись». При этом данный режим доступа к модели продолжает существовать.

Вход

Модель: **sdai_model;**
модель, внутри которой отменяются изменения.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
MO_NEXS СИДД-модель не существует.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».

FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Поведение итератора (**iterator**) на экземпляре агрегата (**aggregation_instance**), созданном, измененном или удаленном после выполнения последней команды начала доступа «чтение—запись» или отмены изменений в настоящем стандарте не определено.

Для последующих экземпляров, созданных после выполнения последней команды начала доступа «чтение—запись» или сохранения изменений в СИДД-модели, должна быть вызвана функция, соответствующая команде удаления:

- **scope;**
- **application_instance;**
- **aggregate_instance,** за исключением **non_persistent_list_instance.**

Для последующих экземпляров, удаленных явно или неявно после выполнения последней команды начала доступа «чтение—запись» или сохранения изменений в СИДД-модели, каждый экземпляр следующих объектов должен быть восстановлен в состоянии, существовавшем до выполнения команды начала доступа «чтение—запись» или сохранения изменений:

- **scope;**
- **application_instance;**
- **aggregate_instance,** за исключением **non_persistent_list_instance.**

Состояния всех следующих экземпляров, измененных после выполнения последней команды начала доступа «чтение—запись» или сохранения изменений в СИДД-модели, должны быть возвращены в исходное положение, существовавшее до выполнения команды начала доступа «чтение—запись» или сохранения изменений:

- **sdai_model,** за исключением **sdai_model.name;**
- **sdai_model_contents;**
- **entity_extent;**
- **scope;**
- **application_instance;**
- **aggregate_instance,** за исключением **non_persistent_list_instance.**

10.7.11 Сохранение изменений

Данная команда фиксирует все изменения содержания, включая **application_instances** и **scopes**, СИДД-модели, внесенные после выполнения последней команды начала доступа «чтение—запись», сохранения или отмены изменений, в зависимости от того, какая команда выполнялась последней. При этом существующий доступ «чтение—запись» остается активным. Команда выполнима только для моделей, к которым разрешен текущий доступ в режиме «чтение—запись». Данная команда обновляет или устанавливает атрибут **change_date** для заданной СИДД-модели.

Вход

Модель: **sdai_model;**
модель, для которой сохраняются изменения.

Указатели возможных ошибок

SS-NOPN Сеанс СИДД не открыт.
MO_NEXS СИДД-модель не существует.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Фиксируются текущие состояния всех экземпляров следующих объектов СИДД-модели:

- **sdai_model_contents;**
- **entity_extent;**
- **scope;**
- **application_instance;**
- **aggregate_instance,** за исключением **non_persistent_list_instance.**

Для текущей даты должен быть установлен атрибут **Model.change_date.**

10.8 Команды области действия

10.8.1 Пополнение области действия

Данная команда добавляет прикладной экземпляр к области действия (**scope**), принадлежащей другому прикладному экземпляру. Если **Target** (Цель) еще не имеет **scope**, создается новая область действия. Это ограничивает область значений допустимых ссылок на **Object** прикладными экземплярами внутри той же области действия.

Вход

Предмет: **application_instance;**
прикладной экземпляр, добавляемый в область действия.

Цель: **application_instance;**
прикладной экземпляр, в область действия которого добавляется **Object**.

Указатели возможных ошибок

EI_NEXS Экземпляр объекта не существует.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Если **Target** еще не имеет области действия, создается новый экземпляр **scope**, принадлежащий СИДД-модели, владеющей **Target**, и для **Target** устанавливается атрибут **scope.owner. Object** добавляется к множеству экземпляров, представленному атрибутом **scope.owned**.

10.8.2 Определение владельца области действия

Данная команда проверяет, обладает ли прикладной экземпляр областью действия.

Вход

Предмет: **application_instance;**
прикладной экземпляр, проверяемый на наличие области действия.

Выход

Результат: **logical_value;**
TRUE, если **Object** обладает областью действия, FALSE, если нет,
UNKNOWN, если область действия не обеспечивается.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
EI_NEXS Экземпляр объекта не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.8.3 Получение области действия

Данная команда возвращает идентификатор области действия, владельцем которой является заданный прикладной экземпляр. Данная команда выдает ошибку SC_NEXS, если **Object** не обладает областью действия.

Вход

Предмет: **application_instance;**
прикладной экземпляр, владеющий областью действия.

Выход

Результат: **scope;**
область действия, владельцем которой является прикладной экземпляр.

Указатели возможных ошибок

SC_NEXS Область действия не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.8.4 Удаление из области действия

Данная команда удаляет прикладной экземпляр из заданной области действия. Если область действия является вложенной внутрь области действия более высокого уровня, тогда прикладной

экземпляр добавляется в следующую область действия более высокого уровня. Если прикладной экземпляр является последним элементом множества **scope.owned**, оставляя область действия без прикладных экземпляров, тогда данная область действия удаляется.

Вход

Предмет: **application_instance;**
прикладной экземпляр, удаляемый из **Target.**

Цель: **scope;**
область действия, владеющая **Object.**

Указатели возможных ошибок

EI_NEXS Экземпляр объекта не существует.
EI_NAVL Экземпляр объекта вне области действия.
SC_NEXS Область действия не существует.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Object должен быть удален из множества **Target.owned**; если **Object** является элементом множества **Target.export_list**, он должен быть удален из данного множества.

Если **Target.owner** принадлежит другой области действия, тогда **Object** должен быть добавлен в множество владения данной области.

Если после удаления **Object** множество **Target.owned** станет пустым, тогда **Target** удаляется.

10.8.5 Добавление к экспортному списку

Данная команда расширяет область значения допустимых ссылок прикладного экземпляра до следующего более высокого уровня. Команда добавляет прикладной экземпляр к экспортному списку области действия. Прикладной экземпляр, экспортируемый из заданной области действия, должен быть элементом множества **Target.owned** или **scope.export_list** вложенной области действия, принадлежащей прикладному экземпляру, являющемуся элементом множества владеемых экземпляров **Target.owned**.

Вход

Предмет: **application_instance;**
экспортируемый прикладной экземпляр.

Цель: **scope;**
область действия, в которую должен быть экспортирован прикладной экземпляр

Указатели возможных ошибок

EI_NEXS Экземпляр объекта не существует.
EI_NAVL Экземпляр объекта вне области действия.
SC_NEXS Область действия не существует.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Object должен быть добавлен к списку **Target.export_list.**

10.8.6 Удаление из экспортного списка

Данная команда восстанавливает ограничение на область допустимых ссылок прикладного экземпляра до уровня, когда эти прикладные экземпляры доступны внутри владеющей области действия. Команда удаляет прикладной экземпляр из экспортного списка области действия.

Вход

Предмет: **application_instance;**
прикладной экземпляр, область допустимых ссылок которого будет ограничена.

Цель: **scope;**
область действия, содержащая прикладной экземпляр в своем экспортном списке.

Указатели возможных ошибок

EI_NEXS Экземпляр объекта не существует.
EI_NAVL Экземпляр объекта вне области действия.
EI_NEXP Экземпляр объекта не экспортируется.
SC_NEXS Область действия не существует.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Object должен быть удален из списка **Target.export_list.**

10.8.7 Удаление области действия

Данная команда удаляет все прикладные экземпляры внутри области действия. Реализация должна вести себя так, как будто команда удаления прикладного экземпляра была выполнена для прикладного экземпляра, владеющего заданной областью действия, и для прикладных экземпляров, принадлежащих к данной области, а затем удалена заданная область действия. Если какие-либо прикладные экземпляры, принадлежащие к области действия, сами владеют областями действия, тогда эти области удаляются подобным же образом. Удаление вложенных областей продолжается до тех пор, пока никакой прикладной экземпляр, принадлежащий к заданной области действия, не будет обладать собственной областью действия.

Вход

Предмет: **scope;**
удаляемая область действия, содержащая прикладные экземпляры.

Указатели возможных ошибок

SC_NEXS Область действия не существует.
TR_NRW Транзакция не имеет типа «чтение—запись».
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Должен быть удален прикладной экземпляр **Object.owner.**

Должны быть удалены прикладные экземпляры, являющиеся элементами множества **Object.owned**.

Должен быть удален **Object**.

Если прикладные экземпляры, принадлежащие к множеству **Object.owned**, являются владельцами собственных областей действия, тогда эти области также удаляются.

10.8.8 Копирование области действия

Данная команда создает новую область действия, копии прикладных экземпляров, владеющих и принадлежащих к заданной области в заданной СИДД-модели, и заполняет атрибуты **scope.owner, scope.owned** и **scope.export_list** на основании скопированных прикладных экземплярах. **TargetModel** может быть СИДД-моделью, внутри которой существует область действия, или другой СИДД-моделью, основанной на той же схеме, что и модель, внутри которой существует данная область. Если какой-либо прикладной экземпляр, принадлежащий к заданной области действия, сам владеет областью действия, тогда эти области копируются подобным же образом. Копирование вложенных

областей продолжается до тех пор, пока не останется прикладных экземпляров, владеющих собственной вложенной областью действия и принадлежащих к заданной области. Все ссылки между прикладными экземплярами в этих областях действия переопределяются для ссылок на вновь созданные копии данных экземпляров.

Вход

Предмет: **scope;**
копируемая область видимости.

ЦелеваяМодель: **sdai_model;**
СИДД-модель, которая должна содержать копируемые область действия и прикладные экземпляры.

Выход

НовыйПредмет: **scope;**
вновь созданная область действия.

Указатели возможных ошибок

SC_NEXS Область действия не существует.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
MO_NEXS СИДД-модель не существует.
MO_NVLD СИДД-модель и область действия основаны на разных схемах.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Новая область действия создается в **TargetModel.**

Прикладной экземпляр **Object.owner** копируется в **TargetModel** и устанавливается в качестве атрибута **NewObject.owner**.

Прикладные экземпляры, являющиеся элементами множества **Object.owned**, копируются в **TargetModel** и устанавливаются в качестве элементов множества **NewObject.owned**.

Все атрибуты вновь созданных прикладных экземпляров со значениями экземпляров объектов переустанавливаются так, чтобы ссылаться на копии исходных прикладных экземпляров внутри новой области действия.

Если любые прикладные экземпляры, являющиеся элементами множества **Object.owned**, владеют собственными областями действия, тогда эти области также копируются в **TargetModel**.

10.8.9 Проверка ссылочных ограничений области действия

Данная команда определяет, какие из ссылочных ограничений всех экземпляров в области действия указанного экземпляра будут удовлетворены. Данная функция проверяет каждый экземпляр в заданной области действия, а также в любой вложенной области.

Вход

Предмет: **application_instance;**
прикладной экземпляр, являющийся владельцем области действия.

Выход

Результат: **logical_value;**
TRUE, если все ограничения удовлетворены, FALSE, если нарушены ограничения ссылок, UNKNOWN, если любое требуемое значение явного атрибута не установлено при ссылке на экземпляр объекта.

Указатели возможных ошибок

EI_NEXS Экземпляр объекта не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.9 Команды типа

10.9.1 Получение определения сложного объекта

Данная команда возвращает созданный тип сложного объекта, составленный из заданных типов объекта и основанный на интерпретации конструкций AND и ANDOR языка EXPRESS

(см. А.1.3). Заданные типы данных объекта должны составлять минимальный (не содержащий повторений) набор конечных типов данных объекта для однозначной идентификации создаваемого типа данных объекта, но могут дополнительно содержать супертипы этих конечных типов данных. Если реализация СИДД не создает тип данных сложного объекта при обработке схемы для заполнения схемы словаря СИДД, данная команда может проверять и создавать новый сложный **entity_definition.**

Вход

Типы: **non_persistent_list_instance;**
список **entity_definition** типов простых объектов, составляющих тип данных сложного объекта.

Выход

Комплекс: **entity_definition;**
конечное определение сложного объекта.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
ED_NDEF Определение объекта не задано.
ED_NVLD Комбинация определения объекта неверна.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Если не существует тип сложного объекта, создается **Complex** — новый экземпляр **entity_definition**. Значения атрибутов **Complex** должны быть установлены согласно приложению А.

10.9.2 Проверка принадлежности к подтипу

Данная команда определяет, является ли тип объекта подтипом другого типа объекта. Отношение подтипа должно быть определено исключительно на основе информации из словаря данных для прикладных схем.

Вход

Тип: **entity_definition;**
проверяемый тип данных объекта.

СравниваемыйТип: **entity_definition;**
потенциально проверяемый супертип.

Выход

Результат: **boolean_value;**
TRUE, если **Type** тот же, что и **CompType** или является его подтипом, или существуют типы А и В, такие, что А эквивалентен по области значений типу В, **Type** является подтипом А, а В — подтипом **CompType**, в противном случае — FALSE.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
ED_NDEF Определение объекта не задано.
ED_NDEQ Определения объектов не принадлежат схемам с эквивалентными областями значений.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.9.3 Проверка принадлежности к подтипу СИДД

Данная команда определяет, является ли тип объекта подтипом другого типа объекта. Отношение подтипа должно быть определено на основе информации из прикладных схем и иерархии экземпляров объектов, установленной в схеме параметризованных данных СИДД (см. раздел 9).

Вход

Тип: **entity_definition;**
проверяемый тип данных объекта.

СравниваемыйТип: **entity_definition;**
потенциально проверяемый супертип.

Выход

Результат: **boolean_value;**

TRUE, если **Type** тот же, что и **CompType** или является его подтипом, или существуют типы A и B, такие, что A эквивалентен по области значений типу B, **Type** является подтипом A, а B — подтипом **CompType**, в противном случае — FALSE.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
ED_NDEF Определение объекта не задано.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.9.4 Проверка эквивалентности областей значений

Данная команда определяет, является ли тип данных объекта эквивалентным по области значений с другим типом данных объекта.

Вход

Тип: **entity_definition;**
проверяемый тип данных объекта.

СравниваемыйТип: **entity_definition;**
тип данных объекта, с которым проводится сравнение.

Выход

Результат: **boolean_value;**
TRUE, если **Type** эквивалентен **CompType** или является его подтипом, или экземпляр **external_schema,** связанный с **schema_definition** собственной схемы **CompType**, определяет **Type** эквивалентным по области значений **CompType**, в противном случае — FALSE.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
ED_NDEF Определение объекта не задано.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.10 Команды экземпляров объектов

10.10.1 Получение значения атрибута

Данная команда возвращает значение атрибута **entity_instance**. Если не существует значения атрибута (потому, что никогда не было ни задано или ни возвращено в неустановленное значение, независимо от того, является ли атрибут необязательным или нет), возвращаемое значение не определяется настоящим стандартом, и должна быть выдана ошибка VA_NSET.

Если интересующий атрибут является **explicit_attribute** с значением экземпляра агрегата, данная команда возвращает идентификатор этого **aggregate_instance**. Если атрибут является вычисляемым (**derived_attribute**) и вводится выражение для его представления, значение вычисляется и возвращается в соответствии с этим выражением. Если результат вычисления является агрегатом (возможно вложенным), тогда должен быть создан нефиксированный список (возможно вложенный) и возвращен его идентификатор. В результате для любого вложенного агрегата должен быть создан нефиксированный список. Если результат вычисления не определен, должна быть выдана ошибка VA_NSET. Если атрибут является инверсным (**inverse_attribute**) и реализация обеспечивает доступ к нему, создается нефиксированный список, содержащий идентификаторы ссылочных экземпляров объектов, и возвращается его идентификатор. Если ссылочные экземпляры объектов отсутствуют, нефиксированный список должен быть пустым. Если не обеспечивается выражение для вычисления или доступ к **derived_attribute**, должна быть выдана ошибка FN_NAVL.

Вход

Предмет: **entity_instance;**
экземпляр объекта, из которого получают значение атрибута.

Атрибут: **attribute;**
атрибут, представляющий интерес.

Выход

Значение: **primitive;**
значение **Attribute в Object.**

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
RP_NOPN	Хранилище не открыто.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EI_NEXS	Экземпляр объекта не существует.
AT_NDEF	Атрибут не определен.
VA_NSET	Явное значение не установлено или оно выведено неопределенно.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.10.2 Проверка атрибута

Данная команда определяет, имеет ли значение атрибут экземпляра объекта. Команда применима только к явным атрибутам.

Примечание — Данная команда применяется независимо от того, объявлен ли атрибут необязательным или нет.

Вход

Предмет: **entity_instance;**
экземпляр объекта, атрибут которого проверяется.

Атрибут: **explicit_attribute;**
проверяемый атрибут.

Выход

Результат: **boolean_value;**
TRUE, если **Attribute** имеет значение в **Object,** FALSE — в противном случае.

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
RP_NOPN	Хранилище не открыто.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EI_NEXS	Экземпляр объекта не существует.
AT_NDEF	Атрибут не определен.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.10.3 Поиск СИДД-модели экземпляра объекта

Данная команда возвращает идентификатор СИДД-модели, содержащей экземпляр объекта.

Вход

Предмет: **entity_instance;**
экземпляр, представляющий интерес.

Выход

Модель: **sdai_model;**
модель, содержащая **Object.**

Указатели возможных ошибок

MO_NEXS	СИДД-модель не существует.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EI_NEXS	Экземпляр объекта не существует.
SY_ERR	Обнаружена ошибка основной системы.

10.10.4 Получение типа экземпляра

Данная команда возвращает идентификатор **entity_definition**, найденный в словаре данных СИДД, на котором основан заданный **entity_instance**.

Вход

Предмет: **entity_instance;**
экземпляр, тип объекта которого ищется.

Выход

Тип: **entity_definition;**
тип объекта, экземпляром которого является **Object.**

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EI_NEXS Экземпляр объекта не существует.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.10.5 Определение соответствия экземпляра заданному типу

Данная команда определяет, является ли **entity_instance** экземпляром точно заданного типа данных объекта, а не одного из его подтипов, или экземпляром типа данных объекта, определенного эквивалентным по области значений с точно заданным типом объекта через экземпляр **domain_equivalent_type** (см. 6.4.8 и приложение А).

Вход

Предмет: **entity_instance;**
проверяемый экземпляр объекта.

Тип: **entity_definition;**
тип объекта, на принадлежность которому проверяется **Object.**

Выход

Результат: **boolean_value;**
TRUE, если тип **Object** является тем же самым, что и **Type,** или если тип **Object** является эквивалентным по области значений **Type,** как это определено экземпляром **domain_equivalent_type** в собственной схеме **Type,** FALSE — в противном случае.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
ED_NDEF Определение объекта не установлено.
EI_NEXS Экземпляр объекта не существует.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.10.6 Определение соответствия экземпляра типу прикладной схемы

Данная команда определяет, является ли **entity_instance** экземпляром конкретного типа или одного из его подтипов, включая случай, когда подтип является составной частью сложного подтипа. Отношение подтипа должно быть определено исключительно на основе информации из прикладных схем.

Вход

Предмет: **entity_instance;**
проверяемый экземпляр объекта.

Тип: **entity_definition;**
тип объекта, на принадлежность которому проверяется **Object.**

Выход

Результат: **boolean_value;**
TRUE, если **Object** является экземпляром того же типа объекта, что и **Type,** или его подтипа, или существуют типы А и В, такие, что А эквивалентен по области значений типу В, а **Object** является экземпляром типа А, игнорирующим эквивалентность области значений, и В является подтипом **Type,** FALSE — в противном случае.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
ED_NDEF Определение объекта не установлено.
EI_NEXS Экземпляр объекта не существует.

FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.10.7 Определение соответствия экземпляра типу прикладной схемы и схемы параметризованных данных СИДД

Данная команда определяет, является ли **entity_instance** экземпляром конкретного типа или одного из его подтипов, включая случай, когда подтип является составной частью сложного подтипа. Отношение подтипа должно быть определено на основе информации из прикладных схем и схемы параметризованных данных СИДД (см. раздел 9).

Вход

Предмет: **entity_instance;**
проверяемый экземпляр объекта.

Тип: **entity_definition;**
тип объекта, на принадлежность которому проверяется **Object.**

Выход

Результат: **boolean_value;**
TRUE, если **Object** является экземпляром того же типа объекта, что и **Type,** или его подтипа, или существуют типы A и B, такие, что A эквивалентен по области значений типу B, а **Object** является экземпляром типа A, игнорирующим эквивалентность области значений, и B является подтипом **Type,**
FALSE — в противном случае.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
ED_NDEF Определение объекта не установлено.
EI_NEXS Экземпляр объекта не существует.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.10.8 Поиск пользователей экземпляра объекта

Данная команда возвращает идентификаторы всех экземпляров объектов, которые ссылаются на заданный экземпляр объекта внутри заданного множества экземпляров схем, и добавляет их к результирующему нефиксированному списку. В случае, когда на заданный экземпляр объекта многократно ссылаются из одного и того же экземпляра объекта, ссылающийся экземпляр должен входить в результирующий список для каждой ссылки.

Вход

Экземпляр: **entity_instance;**
экземпляр объекта, пользователи которого запрашиваются.

Область значений: **non_persistent_list_instance;**
список экземпляров схем (**schema_instances**), устанавливающих области значений экземпляров объектов, проверяемых в качестве пользователей заданного Экземпляра (**Instance**).

Выход

Результат: **non_persistent_list_instance;**
ранее созданный нефиксированный список, в который добавлены идентификаторы экземпляров объектов, ссылающихся на заданный Instance.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
EI_NEXS Экземпляр объекта не существует.
SI_NEXS Экземпляр схемы не существует.
AI_NEXS Экземпляр списка не существует.
RP_NOPN Хранилище не открыто.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Результат (**Result**) должен содержать объекты **entity_instance**, использующие заданный Экземпляр (**Instance**).

10.10.9 Поиск пользователей экземпляра объекта в заданной роли

Данная команда возвращает идентификаторы всех экземпляров объектов, которые ссылаются на заданный экземпляр объекта в заданной роли внутри СИДД-моделей, связанных с заданным множеством экземпляров схем, и добавляет их к результирующему нефиксированному списку. Команда применяется к атрибутам, областью значения которых являются тип объекта заданного экземпляра объекта, любой супертип этого типа или любой определяемый тип, включающий данный тип объекта, или любой супертип данного типа в его исходном типе. В случае, когда на заданный экземпляр объекта в заданной роли многократно ссылаются из одного и того же экземпляра объекта, ссылающийся экземпляр должен возвращаться для каждой ссылки.

Вход

Экземпляр: **entity_instance;**
экземпляр объекта, пользователи которого запрашиваются.

Роль: **attribute;**
атрибут, определяющий требуемую роль.

Область значений: **non_persistent_list_instance;**
список экземпляров схем (**schema_instances**), устанавливающих области значений экземпляров объектов, проверяемых в качестве пользователей заданного Экземпляра (**Instance**).

Выход

Результат: **non_persistent_list_instance;**
ранее созданный нефиксированный список, в который добавлены идентификаторы экземпляров объектов, ссылающихся на заданный **Instance** в заданной Роли (**Role**).

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
EI_NEXS	Экземпляр объекта не существует.
AT_NDEF	Атрибут не определен.
SI_NEXS	Экземпляр схемы не существует.
AI_NEXS	Экземпляр списка не существует.
RP_NOPN	Хранилище не открыто.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Результат (**Result**) должен содержать объекты **entity_instance**, использующие заданный Экземпляр (**Instance**).

10.10.10 Получение значения границы атрибута

Данная команда возвращает текущее значение **population_dependent_bound** действительного, строкового или двоичного типа для значения заданного атрибута в заданном экземпляре объекта. Если существующая совокупность прикладной схемы не является достаточной для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Экземпляр: **entity_instance;**
экземпляр объекта, для которого возвращается значение границы атрибута.

Атрибут: **attribute;**
атрибут Экземпляра (**Instance**), для которого возвращается значение границы атрибута.

Выход

Значение: **integer_value;**
текущее значение границы атрибута.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
VA_NSET Значение границы не установлено.
AT_NDEF Атрибут не определен.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.10.11 Поиск ролей экземпляра

Данная команда возвращает идентификаторы всех атрибутов экземпляров объектов, которые ссылаются на заданный экземпляр объекта внутри заданного множества экземпляров схем, и добавляет их к результирующему нефиксированному списку. В случае, когда на заданный экземпляр объекта многократно ссылаются из одного и того же атрибута, этот атрибут должен входить в результирующий список один раз. В реализациях, обеспечивающих эквивалентность области значений, атрибуты, определенные во внешних схемах, могут быть включены в результирующий список.

Примечание — Данная функция аналогична встроенной функции RolesOf языка EXPRESS (см. ГОСТ Р ИСО 10303-11).

Вход

Экземпляр: **entity_instance;**
экземпляр объекта, пользователи которого запрашиваются.

Область значений: **non_persistent_list_instance;**
список экземпляров схем (**schema_instances**), устанавливающих области значений экземпляров объектов, проверяемых в качестве пользователей заданного Экземпляра (**Instance**).

Выход

Результат: **non_persistent_list_instance;**
ранее созданный нефиксированный список, в который добавлены атрибуты, ссылающиеся на заданный **Instance**.

Указатели возможных ошибок

SS_NOPN Сеанс СИДД не открыт.
EI_NEXS Экземпляр объекта не существует.
SI_NEXS Экземпляр схемы не существует.
AI_NEXS Экземпляр списка не существует.
RP_NOPN Хранилище не открыто.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Результат (**Result**) должен содержать экземпляры атрибутов, использующих заданный **Instance**.

10.10.12 Поиск типов данных экземпляра

Данная команда возвращает идентификаторы всех **named_types**, элементом которых является заданный экземпляр объекта, и добавляет их к результирующему нефиксированному списку. В реализациях, обеспечивающих эквивалентность области значений, типы из внешних схем, эквивалентные по области значений результирующим типам из собственной схемы экземпляра объекта, включаются в результирующий список.

Примечание — Эта функция аналогична встроенной функции TypeOf языка EXPRESS (см. ГОСТ Р ИСО 10303-11).

Вход

Экземпляр: **entity_instance;**
экземпляр объекта, типы которого запрашиваются.

Выход

Результат: **non_persistent_list_instance;**
ранее созданный нефиксированный список, в который добавляются типы Экземпляра (**Instance**).

Указатели возможных ошибок

SS_NOPN	Сеанс СИДД не открыт.
EI_NEXS	Экземпляр объекта не существует.
AI_NEXS	Экземпляр списка не существует.
RP_NOPN	Хранилище не открыто.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Результат (**Result**) должен содержать экземпляры **named_type**, элементом которых является заданный **Instance**.

10.11 Команды прикладного экземпляра

10.11.1 Копирование прикладного экземпляра

Данная команда создает копию заданного прикладного экземпляра в заданной СИДД-модели. Значения атрибутов или элементов агрегатов при любом уровне вложенности в новый экземпляр устанавливаются следующим образом:

- для ссылки на экземпляр объекта новый экземпляр должен ссылаться на те же **application_instances**, что и оригинал;
- для простых типов значения копируются;
- для экземпляров агрегата создаются новые экземпляры агрегата.

ЦелевойМоделью (TargetModel) является СИДД-модель, содержащая копию заданного прикладного экземпляра. Если **TargetModel** не является СИДД-моделью, внутри которой существует **Object**, тогда она должна быть основана по той же схеме, что и СИДД-модель, в которой **Object** существует, или по схеме, содержащей тип объекта, определенный эквивалентным по области значения типу объекта, на котором основан **application_instance.** Значения присваиваются только атрибутам с тем же самым **attribute.name** в эквивалентной области значений типа объекта. В случае различия **TargetModel** и СИДД-модели, в которой существует копируемый **Object**, они должны быть связаны с одним и тем же экземпляром схемы (**schema_instance**).

Примечание — Данная команда, при необходимости, создает ссылки между экземплярами объектов двух различных СИДД-моделей.

Вход

Предмет: **application_instance;**
копируемый прикладной экземпляр.

ЦелеваяМодель: **sdai_model;**
СИДД-модель, которая должна содержать копию **Object**.

Выход

НовыйПредмет: **application_instance;**
вновь созданная копия **Object**.

Указатели возможных ошибок

TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	СИДД-модель недоступна в режиме «чтение—запись».
MX_NDEF	Доступ к СИДД-модели не определен.
MO_NDEQ	СИДД-модель не эквивалентна по области значений прикладному экземпляру.
MO_NEXS	СИДД-модель не существует.
EI_NEXS	Экземпляр объекта не существует.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Для каждого атрибута **Object**, имеющего в качестве значения экземпляр агрегата, **NewObject.values** должен включать новый экземпляр агрегата с тем же содержанием, что и **Object.values**.

Для каждого атрибута **Object**, имеющего в качестве значения экземпляр объекта, **NewObject.values** должен включать тот же **entity_instance**, что и **Object.values**.

Множество **TargetModel.contents.instances** должно содержать **NewObject**.

В экземпляре **entity_extent** из множества **TargetModel.contents.folders**, который имеет значение атрибута **entity_extent.definition** = **Object.definition**, множество **entity_extent.instances** должно содержать **NewObject**. Тот же самый **entity_extent** должен быть в множестве **TargetModel.contents.populated_folders**.

10.11.2 Удаление прикладного экземпляра

Данная команда удаляет прикладной экземпляр. Удаляются все экземпляры агрегатов, созданных как часть этого прикладного экземпляра или вложенные экземпляры агрегатов внутри экземпляра агрегата, созданного как часть удаляемого прикладного экземпляра. После выполнения данной команды любые значения атрибутов прикладных экземпляров, определенные в прикладных или СИДД-схемах, ссылающихся на удаляемый прикладной экземпляр, должны вести себя так, как если бы значения данных атрибутов были не установлены (то есть команда проверки атрибута будет возвращать FALSE, а команда удаления из области действия будет выдавать ошибку EI_NAVL в случае, если удаляемый экземпляр входил в область действия). Любой прикладной экземпляр, на который ссылается удаляемый прикладной экземпляр, не изменяется.

Вход

Предмет: **application_instance;**
удаляемый прикладной экземпляр.

Указатели возможных ошибок

TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	СИДД-модель недоступна в режиме «чтение—запись».
EI_NEXS	Экземпляр объекта не существует.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Object должен быть удален из множества **sdai_model.contents.instances** в СИДД-модели, содержащей данный **Object**.

В экземпляре **entity_extent** из множества **sdai_model.contents.folders**, имеющего значение атрибута **entity_extent.definition** = **Object.definition**, **Object** должен быть удален из множества **entity_extent.instances.**

Если после этого множество **entity_extent.instances** становится пустым, оно должно быть удалено из множества **sdai_model.contents.populated_folders** для СИДД-модели, содержащей **Object**.

10.11.3 Установка значения атрибута

Данная команда присваивает значение явному атрибуту прикладного экземпляра. В случае, когда значением заданного атрибута уже был экземпляр агрегата, данная команда должна вести себя так, как если бы до присвоения атрибуту нового значения была выполнена команда возврата в неустановленное значение.

Вход

Предмет: **application_instance;**
экземпляр, атрибуту которого присваивается значение.

Атрибут: **explicit_attribute;**
атрибут, которому присваивается значение.

Значение: **assignable_primitive;**
новое значение Атрибута (**Attribute**) в **Object**.

Указатели возможных ошибок

TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	СИДД-модель недоступна в режиме «чтение—запись».

EI_NEXS Экземпляр объекта не существует.
AT_NDEF Атрибут не определен.
AT_NVLD Атрибут не является явным атрибутом.
VT_NVLD Тип значения неверен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Значение **Attribute** в **Object** должно иметь заданное Значение (**Value**).

10.11.4 Возврат атрибута в неустановленное значение

Данная команда изменяет состояние заданного атрибута, не обязательного и обязательного, так, чтобы этот атрибут не имел значения в заданном прикладном экземпляре. Последующие команды проверки атрибута будет возвращать значение FALSE. Если значением заданного атрибута уже был экземпляр агрегата, удаляются все экземпляры агрегатов, включая вложенные, связанные с атрибутом в заданном прикладном экземпляре.

Вход

Предмет: **application_instance;**
экземпляр, атрибут которого возвращается в неустановленное значение.

Атрибут: **explicit_attribute;**
атрибут, возвращаемый в неустановленное значение.

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
EI_NEXS Экземпляр объекта не существует.
AT_NDEF Атрибут не определен.
AT_NVLD Атрибут не является явным атрибутом.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Значение **Attribute** в **Object** должно быть возвращено в неустановленное значение.

10.11.5 Создание экземпляра агрегата

Данная команда создает новый, пустой экземпляр агрегата в качестве представления заданного атрибута для прикладного экземпляра, заменяющий любое существующее значение атрибута. Если область значения атрибута является агрегатом агрегатов, создается только самый верхний агрегат. В случае, когда значением заданного атрибута уже был экземпляр агрегата, данная команда должна вести себя так, как если бы до создания нового агрегата была выполнена команда возврата атрибута в неустановленное значение. В случае, когда областью значений заданного атрибута является выбираемый тип (SELECT TYPE) языка EXPRESS, **aggregate_primitive**, обеспечивающий ввод/вывод, должен быть **select_aggregate_instance**, а его атрибут **select_aggregate_instance.data_type** на входе должен быть установлен на значение **defined_type**, определяющее **aggregate_type**, экземпляр которого создает команда. Если команда требует создания экземпляра массива, не являющегося **application_indexed_array_instance**, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы недостаточна для успешного вычисления выражения, определяющего значение индекса для экземпляра массива, то должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и данная реализация не обеспечивает вычисление выражения для индекса массива, должна быть выдана ошибка EX_NSUP.

Вход

Предмет: **application_instance;**
прикладной экземпляр, значение атрибута которого будет установлено.

Атрибут: **explicit_attribute;**
атрибут, значение которого устанавливается.

Вход/Выход

Агрегат: **aggregate_primitive;**
экземпляр агрегата, который будет создан и установлен в качестве значения Атрибута (**Attribute**) в Объекте (**Object**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.

TR_EAB Транзакция прервана аварийно.
MX_NRW СИДД-модель недоступна в режиме «чтение—запись».
EI_NEXS Экземпляр объекта не существует.
AT_NDEF Атрибут не определен.
AT_NVLD Атрибут не является явным атрибутом.
VA_NSET Значение индекса массива не установлено.
EX_NSUP Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Если **Attribute** в качестве своего значения в **Object** прежде имел **aggregate_instance**, то **aggregate_instance** и любой вложенный в него **aggregate_instance** должны быть удалены.

Значением **Attribute** в **Object** должен быть **Aggregate**, являющийся новым пустым агрегатом соответствующего для **Attribute** типа.

10.11.6 Получение постоянной метки

Данная команда возвращает постоянную метку для заданного прикладного экземпляра. Метка должна быть уникальной внутри хранилища, содержащего СИДД-модель, включающую в себя данный прикладной экземпляр. Любой последующий запрос постоянной метки этого же прикладного экземпляра должен возвращать ту же метку в текущем или любом последующем сеансе СИДД.

Вход

Предмет: **application_instance;**
прикладной экземпляр, для которого запрашивается постоянная метка.

Выход

Метка: **string_value;**
постоянная метка **Object.**

Указатели возможных ошибок

TR_NEXS Транзакция не открыта.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
EI_NEXS Экземпляр объекта не существует.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.7 Получение идентификатора сеанса

Данная команда возвращает идентификатор сеанса для прикладного экземпляра, найденного по заданной постоянной метке.

Вход

Метка: **string_value;**
постоянная метка **Object**.

Хранилище: **sdai_repository;**
хранилище, в котором существует выделенный Меткой (**Label**) прикладной экземпляр.

Выход

Предмет: **application_instance;**
прикладной экземпляр, постоянной меткой которого является **Label**.

Указатели возможных ошибок

TR_NEXS Транзакция не открыта.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
RP_NEXS Хранилище не существует.
EI_NEXS Экземпляр объекта не существует.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.8 Получение описания

Данная команда возвращает воспринимаемое человеком описание заданного прикладного экземпляра. Любой последующий запрос описания того же прикладного экземпляра должен возвра-

щать то же описание в текущем сеансе СИДД. Для реализаций, в которых прикладные экземпляры существуют в файлах, закодированных согласно ГОСТ Р ИСО 10303-21, форма описания должна быть следующей: имя прикладного экземпляра, затем пробел, потом имя файла, содержащего прикладной экземпляр.

Вход

Предмет: **application_instance;**
прикладной экземпляр, для которого запрашивается описание.

Выход

Метка: **string_value;**
описание **Object**.

Указатели возможных ошибок

TR_NEXS Транзакция не открыта.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
RP_NOPN Хранилище не открыто.
EI_NEXS Экземпляр объекта не существует.
SY_ERR Обнаружена ошибка основной системы.

10.11.9 Проверка правила «were»

Данная команда проверяет, удовлетворено ли правило «were» для заданного прикладного экземпляра. Это правило может быть объявлено явно как часть **entity_definition**, на котором основан прикладной экземпляр, или в **defined_type**, ограничивающем значение атрибута, объявленного выше в **entity_definition**.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Правило: **where_rule;**
правило «where», проверяемое для **Object**.

Выход

Результат: **logical_value;**
TRUE, если Правило (**Rule**) удовлетворено, FALSE, если **Rule** нарушено, и UNKNOWN, если значение вычисленного выражения не определено или равно UNKNOWN.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
RU_NEXS Правило не определено.
EI_NEXS Экземпляр объекта не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения индекса не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.10 Проверка наличия значений у явных атрибутов

Данная команда определяет, имеют ли значения все требуемые явные атрибуты прикладного экземпляра.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, не соответствующие проверяемым требованиям.

Выход

Результат: **boolean_value;**
TRUE, если все обязательные атрибуты Предмета (**Object**) имеют значения или **Object** не имеет обязательных атрибутов, FALSE, если какой-либо обязательный атрибут не имеет значения в **Object**.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
EI_NEXS Экземпляр объекта не существует.
AI_NEXS Список не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.11 Проверка инверсных атрибутов

Данная команда определяет, все ли ограничения количества элементов, заданные в объявлениях инверсных атрибутов, удовлетворены в прикладном экземпляре.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры инверсных атрибутов (**inverse_attribute**), элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход

Результат: **boolean_value;**
TRUE, если для Предмета (**Object**) удовлетворены все ограничения инверсного (INVERS) атрибута или **Object** не имеет инверсных атрибутов, FALSE, если какие-либо ограничения инверсного атрибута нарушены.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
EI_NEXS Экземпляр объекта не существует.
AI_NEXS Список не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.12 Проверка ссылок явных атрибутов

Данная команда определяет, все ли экземпляры объектов, являющиеся значениями атрибутов прикладного экземпляра, имеют допустимый для этих атрибутов тип данных объекта.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход

Результат: **logical_value;**
TRUE, если нет атрибутов Предмета (**Object**), являющихся экземплярами объекта, имеющими значения неверного типа, или **Object** не имеет атрибутов, ссылающихся на экземпляры объектов, FALSE, если какой-либо атрибут имеет значение, являющееся экземпляром объекта неверного типа, UNKNOWN, если значение какого-либо обязательного явного атрибута, являющееся ссылкой на экземпляр объекта, не установлено.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
EI_NEXS Экземпляр объекта не существует.
AI_NEXS Список не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.

TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.13 Проверка размерности агрегата

Данная команда определяет, удовлетворяет ли число элементов или значений допустимых индексов для экземпляров массива любых атрибутов заданного прикладного экземпляра ограничениям, установленным в объявленных типах этих атрибутов. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Предмет: **application_instance;**
экземпляр, представляющий интерес.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход

Результат: **logical_value;**
TRUE, если удовлетворены все ограничения размера агрегата для экземпляров агрегатов, имеющих значения атрибутов Предмета (**Object**), или **Object** не имеет значений атрибутов, являющихся экземплярами агрегата, FALSE, если по крайней мере одно ограничение размера нарушено, UNKNOWN, если вычисленное значение выражения границы не определено или равно UNKNOWN.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
VA_NSET Значение границы не установлено.
EI_NEXS Экземпляр объекта не существует.
AI_NEXS Список не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.11.14 Проверка уникальности агрегатов

Данная команда определяет уникальность (однозначность) всех элементов в любом экземпляре агрегата, являющимся значением какого-либо атрибута, объявленный тип которого требует наличия уникальности. Эту проверку проводят для всех атрибутов конкретного экземпляра. Уникальность определяется сравнением идентификаторов экземпляров (см. 12.2.2 ГОСТ Р ИСО 10303-11) для случаев, когда экземпляры объекта являются элементами агрегата. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход

Результат: **boolean_value;**
TRUE, если все ограничения уникальности удовлетворены или Предмет (**Object**) не имеет экземпляров агрегата в качестве значений атрибута, FALSE, если по крайней мере одно ограничение уникальности нарушено.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
EI_NEXS	Экземпляр объекта не существует.
AI_NEXS	Список не существует.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
VA_NSET	Значение границы не установлено.
EX_NSUP	Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.11.15 Проверка массива на наличие пустых элементов

Данная команда проверяет, имеют ли значения все индексные позиции экземпляров массива, объявленный тип которого не допускает наличия пустых элементов. Проверку проводят для всех атрибутов заданного прикладного экземпляра, имеющих экземпляры массива в качестве своих значений. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход

Результат: **boolean_value;**
TRUE, если все экземпляры массивов, не могущие иметь позиций с неустановленными значениями, фактически имеют значения во всех индексных позициях или все экземпляры массивов, представляющие атрибуты Предмета (**Object**), объявлены содержащими необязательные элементы, или **Object** не имеет атрибутов со значениями экземпляров массива, FALSE, если отсутствует по крайней мере одно значение атрибута экземпляра массива, объявленного не содержащим необязательные элементы, или нижняя или верхняя граница экземпляра массива конфликтует с объявлением типа массива.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
VA_NSET	Значение индекса массива не установлено.
EI_NEXS	Экземпляр объекта не существует.
AI_NEXS	Список не существует.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EX_NSUP	Вычисление выражения индекса не обеспечивается данной реализацией.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.11.16 Проверка ширины строки

Данная команда проверяет, будут ли строки с данными значениями атрибутов иметь заданную ширину. Проверку проводят для всех атрибутов конкретного экземпляра с соответствующим строковым значением. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение ширины, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход
Результат: **logical_value;**
TRUE, если все строковые значения атрибутов имеют правильную ширину, FALSE, если по крайней мере одно строковое значение атрибута нарушает объявленную ширину, UNKNOWN, если значение вычисленного выражения для вычисляемого атрибута не определено.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
VA_NSET	Значение границы массива не установлено.
EI_NEXS	Экземпляр объекта не существует.
AI_NEXS	Список не существует.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EX_NSUP	Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.11.17 Проверка ширины двоичного значения

Данная команда проверяет, имеют ли двоичные (BINARY) значения атрибутов заданную ширину. Проверку проводят для всех двоичных значений атрибутов отдельного экземпляра. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение ширины, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход
Предмет: **application_instance;**
проверяемый экземпляр.
Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход
Результат: **logical_value;**
TRUE, если все двоичные значения атрибутов имеют правильную ширину, FALSE, если по крайней мере одно двоичное значение атрибута нарушает объявленную ширину, UNKNOWN, если значение вычисленного выражения для вычисляемого атрибута не определено.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
VA_NSET	Значение границы не установлено.
EI_NEXS	Экземпляр объекта не существует.
AI_NEXS	Список не существует.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EX_NSUP	Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.11.18 Проверка точности действительного значения

Данная команда проверяет, имеют ли действительные значения атрибутов заданную минимальную точность. Проверку проводят для всех атрибутов отдельного экземпляра, имеющих действительные значения. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение точности, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Предмет: **application_instance;**
проверяемый экземпляр.

Несоответствия: **non_persistent_list_instance;**
если Результат (**Result**) FALSE, то это существующий нефиксированный список, в который добавляются экземпляры атрибутов, элементами которых являются атрибуты, не соответствующие проверяемым требованиям.

Выход

Результат: **logical_value;**
TRUE, если все действительные значения атрибутов имеют по крайней мере объявленную точность, FALSE, если по крайней мере одно действительное значение атрибута нарушает объявленную точность, UNKNOWN, если значение вычисленного выражения для вычисляемого атрибута не определено.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
VA_NSET Значение границы не установлено.
EI_NEXS Экземпляр объекта не существует.
AI_NEXS Список не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.12 Команды агрегата экземпляров объекта

10.12.1 Получение числа элементов

Данная команда возвращает число элементов, содержащихся в экземпляре агрегата, или, если экземпляр агрегата является экземпляром массива, — размерность экземпляра массива.

Вход

Агрегат: **aggregate_instance;**
подсчитываемый агрегат.

Выход

Результат: **integer_value;**
текущее число элементов в экземпляре Агрегата (**Aggregate**) или, если данный экземпляр является экземпляром массива, — его размерность.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

10.12.2 Проверка на вхождение в экземпляр агрегата

Данная команда проверяет, является ли заданное значение элементом конкретного экземпляра агрегата.

Вход

Агрегат: **aggregate_instance;**
проверяемый экземпляр агрегата.

Значение: **primitive;**
проверяемое значение.

Выход

Результат: **boolean_value;**
TRUE, если значение содержится в агрегате, в противном случае — FALSE.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.

TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

10.12.3 Создание итератора

Данная команда создает новый итератор для экземпляра агрегата. Итератор первоначально назначается так, как если бы была выполнена команда установки в начальное положение таким образом, что нет ни одного текущего элемента.

Вход

Агрегат: **aggregate_instance;**
экземпляр агрегата, для которого создается новый итератор.

Выход

Итератор: **iterator;**
вновь созданный итератор для агрегата.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Итератор (**Iterator**) должен быть вновь установленным для Агрегата (**Aggregate**).
Iterator должен быть установлен в начале **Aggregate** и не должен иметь текущего элемента.

10.12.4 Удаление итератора

Данная команда удаляет существующий итератор.

Вход

Итератор: **iterator;**
удаляемый итератор.

Указатели возможных ошибок

IR_NEXS Итератор не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Iterator больше не должен существовать.

10.12.5 Установка в начальное положение

Данная команда устанавливает итератор в начале соответствующего экземпляра агрегата так, чтобы отсутствовал текущий элемент. Для упорядоченных коллекций (наборов) данная команда устанавливает итератор так, чтобы команда на переход к следующему элементу сделала первый элемент экземпляра агрегата текущим элементом. Для неупорядоченных коллекций эта команда восстанавливает отслеживание реализацией того, какие элементы будут пройдены командой перехода к следующему элементу (см. 10.12.6). Для неупорядоченных коллекций определение первого или следующего элемента оставлено на усмотрение реализации.

Вход

Итератор: **iterator;**
итератор, устанавливаемый в начальное положение.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
IR_NEXS Итератор не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Итератор (**Iterator**) должен быть установлен в начале агрегата **Iterator.subject** так, чтобы отсутствовал текущий элемент.

10.12.6 Переход к новому текущему элементу

Данная команда устанавливает итератор так, чтобы он содержал новый текущий элемент. Итератор должен вести себя следующим образом:

- если он установлен в начале соответствующего экземпляра агрегата, данная команда делает первый элемент данного экземпляра текущим;

- если он был установлен на последний элемент соответствующего экземпляра агрегата, в конце данного экземпляра или экземпляр был пустой, то итератор будет переустановлен в конец экземпляра агрегата и не будет иметь никакого текущего элемента;

- если связанный с итератором экземпляр агрегата является упорядоченным набором (коллекцией), новым текущим элементом будет элемент, непосредственно следующий за установленным, если он не является последним. В этом случае итератор будет установлен командой на переход к предыдущему элементу, которая текущим сделает последний элемент соответствующего экземпляра агрегата;

- если связанный с итератором экземпляр агрегата является неупорядоченным набором (коллекцией), реализация должна отслеживать просмотр элементов так, чтобы многократные вызовы команды на переход к следующему элементу перемещали итератор по всем элементам соответствующего экземпляра агрегата без повторений. Определение того, какой из оставшихся элементов будет следующим, оставлено на усмотрение реализации.

Вход

Итератор: **iterator;**
устанавливаемый итератор.

Выход

Результат: **boolean_value;**
TRUE, если Итератор (**Iterator**) установлен с новым текущим элементом,
FALSE, если **Iterator** не установлен с новым текущим элементом, так как не существует последующего элемента агрегата.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
AI_NEXS	Экземпляр агрегата не существует.
IR_NEXS	Итератор не существует.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Iterator должен быть установлен с текущим элементом, непосредственно следующим за предыдущим текущим элементом, существовавшим перед вызовом данной команды, как это описано выше.

Iterator.position увеличивается.

10.12.7 Получение текущего элемента

Данная команда возвращает значение текущего элемента, на который указывает итератор.

Вход

Итератор: **iterator;**
итератор, определяющий экземпляр агрегата и возвращаемый элемент.

Выход

Значение: **primitive;**
текущий член, указанный итератором.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
AI_NEXS	Экземпляр агрегата не существует.
AL_NSET	Экземпляр агрегата является пустым.
IR_NEXS	Итератор не существует.
IR_NSET	Текущий элемент в итераторе не установлен.
VA_NSET	Значение позиции текущего элемента агрегата не установлено.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
SY_ERR	Обнаружена ошибка основной системы.

10.12.8 Получение значения границы по итератору

Данная команда возвращает значение **population_dependent_bound** действительного, строкового или двоичного типа для значения элемента агрегата, на который указывает итератор. Если суще-

ствующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Итератор: **iterator;**
итератор, определяющий значение элемента агрегата, для которого возвращается значение границы.

Выход

Значение: **integer_value;**
значение граничного значения элемента агрегата.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
VA_NSET Значение границы связанной совокупности не установлено.
IR_NEXS Итератор не существует.
AI_NEXS Экземпляр агрегата не существует.
VT_NVLD Тип значения границы не зависит от совокупности.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.12.9 Получение нижней границы

Данная команда возвращает значение **population_dependent_bound** для нижней границы или нижнего индекса заданного экземпляра агрегата, основанного на текущей совокупности прикладной схемы. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Агрегат: **aggregate_instance;**
экземпляр агрегата, для которого возвращается значение нижней границы.

Выход

Значение: **integer_value;**
значение нижней границы или нижнего индекса.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
VA_NSET Значение границы связанной совокупности не установлено.
VT_NVLD Тип значения границы не зависит от совокупности.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.12.10 Получение верхней границы

Данная команда возвращает значение **population_dependent_bound** для верхней границы или верхнего индекса заданного экземпляра агрегата, основанного на текущей совокупности прикладной схемы. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Агрегат: **aggregate_instance;**
экземпляр агрегата, для которого возвращается значение верхней границы.

Выход

Значение: **integer_value;**
значение верхней границы или верхнего индекса.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
AI_NEXS	Экземпляр агрегата не существует.
VA_NSET	Значение границы связанной совокупности не установлено.
VT_NVLD	Тип значения границы не зависит от совокупности.
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
EX_NSUP	Вычисление выражения границы не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.13 Команды агрегата прикладных экземпляров

10.13.1 Создание экземпляра агрегата в качестве текущего элемента

Данная команда заменяет значение текущего элемента экземпляра агрегата новым, пустым экземпляром агрегата. Если ранее существовавшее значение было экземпляром агрегата (**aggregate_instance**), оно уничтожается вместе со всеми вложенными в него **aggregate_instance**. Новый экземпляр агрегата заменяет предыдущее значение в качестве текущего элемента для заданного итератора. Если областью значений агрегата, связанного с заданным итератором, является SELECT TYPE языка EXPRESS, **aggregate_primitive**, используемый в качестве ввода/вывода, должен быть **select_aggregate_instance**, а **select_aggregate_instance.data_type** на входе должен быть установлен со значением **defined_type**, задающим **aggregate_type**, экземпляр которого создает операция. Если операция требует создание экземпляра массива, который не является **application_indexed_array_instance**, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение индекса экземпляра массива, должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и реализация не обеспечивает вычисление выражения для индекса массива, должна быть выдана ошибка EX_NSUP.

Вход

Итератор: **iterator;**
итератор, определяющий агрегат и замещаемый текущий элемент.

Вход/Выход

НовыйАгрегат: **aggregate_primitive;**
новый экземпляр агрегата, создаваемый в качестве текущего элемента.

Указатели возможных ошибок

TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS	Экземпляр агрегата не существует.
IR_NEXS	Итератор не существует.
IR_NSET	Текущий элемент в итераторе не установлен.
VA_NSET	Значение индекса экземпляра массива связанной совокупности не установлено.
EX_NSUP	Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

НовыйАгрегат (**NewAggregate**) должен быть добавлен к содержимому **Iterator.subject**, заменяя предыдущее значение **Iterator.current_memder**.

Устанавливается **Iterator.current_member** для ссылки на **NewAggregate**.

Если предыдущий текущий элемент был экземпляром агрегата (**aggregate_instance**), он удаляется. Любой **aggregate_instance**, являющийся вложенным элементом заменяемого **aggregate_instance**, также удаляется.

10.13.2 Задание значения текущего элемента

Данная команда заменяет значение текущего элемента экземпляра агрегата заданным значением. Если существовавшим значением был **aggregate_instance**, он удаляется вместе со всеми вложенными в него **aggregate_instance**. Новое значение заменяет ранее существовавшее в качестве текущего элемента заданного итератора.

Вход

Итератор: **iterator;**
итератор, определяющий элемент изменяемого экземпляра агрегата.

Значение: **assignable_primitive;**
значение, присваиваемое соответствующему элементу Итератором (**Iterator**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
VT_NVLD Тип значения неверен.
AI_NEXS Экземпляр агрегата не существует.
AI_NSET Экземпляр агрегата пустой.
IR_NEXS Итератор не существует.
IR_NSET Текущий элемент в итераторе не установлен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Значение (**Value**) должно быть добавлено к содержанию **Iterator.subject**, заменяющему ранее существовавшее значение **Iterator.current_member**.

Iterator.current_member устанавливается ссылкой на **Value**.

Если существовавшим текущим элементом был экземпляром агрегата (**aggregate_instance**), он удаляется. Любой **aggregate_instance**, являющийся вложенным элементом удаляемого экземпляра, также удаляется.

10.13.3 Удаление текущего элемента

Данная команда удаляет текущий элемент агрегата, указанный итератором. Итератор затем устанавливается так, как будто бы перед удалением элемента была выполнена команда перехода к следующему элементу. Данная команда не может быть применена для **array_instance**. Если указанным элементом был **aggregate_instance**, он удаляется вместе со всеми вложенными в него **aggregate_instance**.

Вход

Итератор: **iterator;**
итератор, определяющий удаляемый агрегат и его элемент.

Выход

Результат: **boolean_value;**
TRUE, если Итератор (**Iterator**) установлен с новым текущим элементом,
FALSE, если **Iterator** не установлен с новым текущим элементом, так как не существует последующего элемента.

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата является массивом.
IR_NEXS Итератор не существует.
IR_NSET Текущий элемент в итераторе не установлен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Iterator.current_member должен быть удален из **Iterator.subject.**

Iterator.current_member устанавливается так, как будто бы перед удалением существующего элемента была выполнена команда перехода к следующему элементу.

10.14 Команды неупорядоченного набора (коллекции) прикладных экземпляров

10.14.1 Неупорядоченное добавление

Данная команда добавляет значение в качестве элемента неупорядоченного набора (коллекции).

Вход

Агрегат: **unordered_collection;**
набор (set) или мультимножество (bag), в которое добавляется Значение (**Value**).

Значение: **assignable_primitive;**
значение, добавляемое в Агрегат (**Aggregate**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является неупорядоченным набором (коллекцией).
VT_NVLD Тип значения неверен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Value (Значение) должно быть добавлено в качестве элемента множества **Aggregate.contents**.

10.14.2 Неупорядоченное создание экземпляра агрегата

Данная команда добавляет новый, пустой экземпляр агрегата в качестве элемента существующего неупорядоченного набора (коллекции). Если областью значений заданного экземпляра агрегата является выбираемый тип (SELECT TYPE) языка EXPRESS, **aggregate_primitive**, используемый в качестве ввода/вывода, должен быть **select_aggregate_instance**, а для определения типа (TYPE) экземпляра, создаваемого командой, на входе должен быть установлен атрибут **select_aggregate_instance.data_type**. Если операция требует создания экземпляра массива, который не является **application_indexed_array_instance**, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение индекса экземпляра массива, должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и реализация не обеспечивает вычисление выражения для индекса массива, должна быть выдана ошибка EX_NSUP.

Вход

Агрегат: **unordered_collection;**
экземпляр агрегата, в который добавляется значение.

Вход/Выход

НовыйАгрегат: **aggregate_instance;**
новый экземпляр агрегата, который добавляется к Агрегату (**Aggregate**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является неупорядоченным набором (коллекцией).
VA_NSET Значение индекса экземпляра массива связанной совокупности не установлено.
EX_NSUP Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

НовыйАгрегат (**NewAggregate**) должен быть новым пустым экземпляром агрегата, соответствующим типу элемента Агрегата (**Aggregate**).

NewAggregate должен быть добавлен в качестве элемента множества **Aggregate.contents**.

10.14.3 Неупорядоченное удаление

Данная команда удаляет одно вхождение заданного значения из содержания неупорядоченного набора (коллекции). Если удаляемый элемент является экземпляром агрегата, он и любые вложенные в него экземпляры агрегата удаляются.

Вход

Агрегат: **unordered_collection;**
набор (set) или мультимножество (bag), из которого удаляется Значение (**Value**).

Значение: **primitive;**
значение, удаляемое из Агрегата (**Aggregate**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является неупорядоченным набором (коллекцией).
VA_NEXS Значение не существует в экземпляре агрегата.
VT_NVLD Тип значения неверен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Value должно быть удалено как элемент множества **Aggregate.contents**.

10.15 Команды упорядоченного набора (коллекции) экземпляров объектов

10.15.1 Получение по индексу

Данная команда возвращает значение элемента из конкретной позиции индекса в упорядоченном наборе (коллекции).

Вход

Агрегат: **ordered_collection;**
массив или список, из которого возвращается значение.

Индекс: **integer_value;**
индекс массива или позиция в списке, из которой возвращается значение.

Выход

Значение: **primitive;**
значение из позиции Индекса (**Index**) в Агрегате (**Aggregate**).

Указатели возможных ошибок

MX_NDEF Доступ к СИДД_модели не определен.
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является упорядоченным набором (коллекцией).
IX_NVLD Позиция индекса неверна.
VT_NVLD Тип значения не является целым.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

10.15.2 Переход в конец

Данная команда устанавливает итератор над упорядоченной коллекцией так, что в нем не будет определен текущий элемент, а выполнение команды перехода к предыдущему элементу приведет к тому, что текущим станет последний элемент упорядоченного набора (коллекции).

Вход

Итератор: **iterator;**
устанавливаемый итератор.

Указатели возможных ошибок

AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является упорядоченным набором (коллекцией).
IR_NEXS Итератор не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Итератор (**Iterator**) должен быть установлен в конец агрегата **Iterator.subject** так, чтобы в нем отсутствовал текущий элемент.

10.15.3 Переход к предыдущему элементу

Данная команда устанавливает предыдущий элемент упорядоченного набора (коллекции) в качестве нового текущего элемента. Если итератор был установлен в конце соответствующего экземпляра агрегата и в нем отсутствует текущий элемент, данная команда устанавливает в качестве текущего последний элемент данного экземпляра. Если итератор был установлен на первом элементе соответствующего экземпляра агрегата или в начале данного экземпляра, или экземпляр является пустым, итератор устанавливается в начале экземпляра агрегата и в нем отсутствует текущий элемент.

Вход

Итератор: **iterator;**
устанавливаемый итератор.

Выход

Результат: **boolean_value;**
TRUE, если существует элемент в новой текущей позиции, FALSE, если итератор был установлен в начале агрегата.

Указатели возможных ошибок

AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является упорядоченным набором (коллекцией).
IR_NEXS Итератор не существует.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Итератор (**Iterator**) должен быть установлен так, чтобы его текущим элементом стал элемент, непосредственно предшествовавший текущему элементу, существовавшему ранее перед вызовом данной команды (как описано выше).

10.15.4 Получение значения границы по индексу

Данная команда возвращает значение **population_dependent_bound** для действительного, строкового или двоичного типа значения элемента агрегата в заданной позиции индекса. Если существующая совокупность прикладной схемы не является достаточной для успешного вычисления выражения, определяющего значение границы, должна быть выдана ошибка VA_NSET. Если реализация не обеспечивает вычисление данного выражения, должна быть выдана ошибка EX_NSUP.

Вход

Агрегат: **ordered_collection;**
экземпляр агрегата, содержащий значение элемента, для которого возвращается значение границы.

Индекс: **integer_value;**
позиция индекса, определяющего элемент агрегата, для которого возвращается значение границы.

Выход

Значение: **integer_value;**
величина значения границы элемента агрегата.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
VA_NSET Значение границы агрегата связанной совокупности не установлено.
IX_NVLD Позиция индекса неверна.
VT_NVLD Тип значения неверен.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения границы не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.16 Команды упорядоченного набора (коллекции) прикладных экземпляров

10.16.1 Внесение значения по индексу

Данная команда заменяет значение элемента в конкретной индексной позиции упорядоченного набора (коллекции). Если предыдущее значение было экземпляром агрегата (**aggregate_instance**), оно удаляется вместе с любыми вложенными в него **aggregate_instance**.

Вход

Агрегат: **ordered_collection;**
изменяемый массив или список.

Индекс: **integer_value;**
индекс массива или позиция в списке для данного элемента.

Значение: **assignable_primitive;**
значение, устанавливаемое в Агрегате (**Aggregate**) для Индекса (**Index**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является упорядоченным набором (коллекцией).
IX_NVLD Позиция индекса неверна.
VT_NVLD Тип значения неверен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Элементом **Aggregate** в позиции **Index** должно быть **Value**.

10.16.2 Создание экземпляра агрегата по индексу

Данная команда заменяет ранее существующий элемент упорядоченного набора (коллекции) пустым экземпляром агрегата. Заменяемый элемент определяется своим целочисленным индексом позиции внутри упорядоченного набора. Если ранее существовавший элемент был экземпляром агрегата, он уничтожается вместе со всеми вложенными в него экземплярами агрегатов (**aggregate_instance**). Если областью значений заданного экземпляра агрегата является выбираемый тип (SELECT TYPE) языка EXPRESS, **aggregate_primitive**, используемым в качестве ввода/вывода, должен быть **select_aggregate_instance**, а для определения типа (TYPE) экземпляра, создаваемого командой, на входе должен быть установлен атрибут **select_aggregate_instance.data_type**. Если операция требует создания экземпляра массива, который не является **application_indexed_array_instance**, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы недостаточна для успешного вычисления выражения, определяющего значение индекса экземпляра массива, должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и реализация не обеспечивает вычисление выражения для индексов массива, должна быть выдана ошибка EX_NSUP.

Вход

Агрегат: **ordered_collection;**
экземпляр агрегата, содержащего заменяемый элемент.

Индекс: **integer_value;**
индекс массива или позиция в списке для заменяемого элемента.

Выход

НовыйАгрегат: **aggregate_primitive**;
новый экземпляр агрегата, добавляемый в позицию Агрегата (**Aggregate**), заданную Индексом (**Index**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является упорядоченным набором (коллекцией).
IX_NVLD Позиция индекса неверна.
VA_NSET Значение индекса экземпляра массива соответствующей совокупности не установлено.
EX_NSUP Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

НовыйАгрегат (**NewAggregate**) должен быть пустым экземпляром агрегата, имеющим тип соответствующего элемента Агрегата (**Aggregate**).

NewAggregate должен быть элементом **Aggregate** в позиции, установленной **Index**.

10.17 Команды массива экземпляров объекта

10.17.1 Проверка наличия значения по индексу

Данная команда проверяет, установлено ли значение элемента в конкретной индексной позиции экземпляра массива.

Вход

Агрегат: **array_instance;**
проверяемый экземпляр массива.

Индекс: **integer_value;**
индексная позиция, проверяемая в Агрегате (**Aggregate**).

Выход

Результат: **boolean_value;**
TRUE, если значение установлено в позиции Индекса (**Index**) Агрегата (**Aggregate**), FALSE — в противном случае.

Указатели возможных ошибок

AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом.
IX_NVLD Позиция индекса неверна.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

10.17.2 Проверка текущего элемента

Данная команда определяет, установлено ли значение элемента в конкретной индексной позиции экземпляра массива, заданной итератором.

Вход

Итератор: **iterator;**
итератор, задающий проверяемую позицию массива.

Выход

Результат: **boolean_value;**
TRUE, если значение установлено в позиции, заданной Итератором (**Iterator**), FALSE — в противном случае.

Указатели возможных ошибок

AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом.
IR_NEXS Итератор не существует.
IR_NSET В итераторе не установлен текущий элемент.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
SY_ERR Обнаружена ошибка основной системы.

10.17.3 Получение нижнего индекса

Данная команда возвращает значение **population_dependent_bound** для нижнего индекса заданного экземпляра массива, основанного на совокупности прикладной схемы, существовавшей при создании этого экземпляра, или соответствующее значение, измененное последней командой переиндексирования массива или сброса индексов массива для заданного экземпляра.

Вход

Массив: **array_instance;**
экземпляр массива, для которого возвращается значение нижнего индекса.

Выход

Значение: **integer_value;**
значение нижнего индекса.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом соответствующей совокупности.

TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.17.4 Получение верхнего индекса

Данная команда возвращает значение **population_dependent_bound** для верхнего индекса заданного экземпляра массива, основанного на совокупности прикладной схемы, существовавшей при создании этого экземпляра, или соответствующее значение, измененное последней командой переиндексирования массива или сброса индексов массива для заданного экземпляра.

Вход

Массив: **array_instance;**
экземпляр массива, для которого возвращается значение верхнего индекса.

Выход

Значение: **integer_value;**
значение верхнего индекса.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом соответствующей совокупности.
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.18 Команды массива прикладных экземпляров

10.18.1 Возврат к неустановленному значению по индексу

Данная команда изменяет значение элемента заданного экземпляра массива в заданной позиции так, чтобы экземпляр массива не имел значения в этой позиции. Позиция задается индексом. Последующие команды проверки значения в данной позиции должны возвращать FALSE. Если предыдущее значение было экземпляром агрегата (**aggregate_instance**), оно удаляется вместе со всеми вложенными **aggregate_instance**.

Вход

Агрегат: **array_instance;**
изменяемый массив.

Индекс: **integer_value;**
индекс массива, по которому элемент массива возвращается к неустановленному значению.

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом.
IX_NVLD Позиция индекса неверна.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Элемент Агрегата (**Aggregate**), указанный Индексом (**Index**), должен быть возвращен к неустановленному значению.

10.18.2 Возврат текущего элемента к неустановленному значению

Данная команда изменяет значение элемента заданного экземпляра массива в заданной позиции так, чтобы экземпляр массива не имел значения в этой позиции. Позиция задается итератором. Последующие команды проверки значения в данной позиции должны возвращать FALSE. Если предыдущее значение было экземпляром агрегата (**aggregate_instance**), оно удаляется вместе со всеми вложенными **aggregate_instance**. Данная команда может применяться только к итератору экземпляра массива.

Вход

Итератор: **iterator;**

итератор, задающий экземпляр массива и позицию в нем, по которой элемент массива возвращается к неустановленному значению.

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом.
IR_NEXS Итератор не существует.
IR_NSET Итератор не имеет текущего элемента.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Текущий элемент массива **Iterator.subject**, указанный Итератором (**Iterator**), должен быть возвращен к неустановленному значению.

10.18.3 Переиндексирование массива

Данная команда изменяет размер нижнего или верхнего индекса (или обоих), базирующихся на значении **population_dependent_bound**, для заданного экземпляра массива, основанного на текущей совокупности прикладной схемы. После успешного выполнения данной команды размер заданного экземпляра массива базируется на новых значениях индексов, а правильными являются только позиции массива, связанные с новыми значениями индекса. Данная команда не влияет на любые элементы массива в индексных позициях, оставшихся правильными при новых значениях индексов. Недоступными становятся любые элементы массива в индексных позициях, ставших неверными при новых значениях индексов. Если значением такого недоступного элемента массива был экземпляр агрегата, он удаляется вместе с любыми вложенными в него экземплярами агрегатов. Любые новые индексные позиции элементов массива, полученные в результате данной команды, будут иметь неустановленные значения. Если существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение индекса, должна быть выдана ошибка VA_NSET.

Вход

Агрегат: **array_instance;**

переиндексируемый экземпляр массива.

Указатели возможных ошибок

MX_NDEF Доступ к СИДД-модели не определен.
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является массивом соответствующей совокупности.
VA_NSET Значение индекса экземпляра массива соответствующей совокупности не установлено.
TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
EX_NSUP Вычисление выражения индекса не обеспечивается данной реализацией.
FN_NAVL Функция не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

10.18.4 Переустановка индексов массива

Данная команда переустанавливает значения верхнего и нижнего индексов для заданного экземпляра массива. Значения индексов этого экземпляра должны быть основаны на **population_dependent_bound**. После успешного выполнения данной команды размер заданного экземпляра массива основывается на новых значениях индексов, а правильными являются только позиции массива, соответствующие новым значениям индексов. Данная команда не влияет на любые элементы массива в индексных позициях, оставшихся верными при новых значениях индексов. Недоступными становятся любые элементы массива в индексных позициях, ставших ошибочными при новых значениях индексов. Если значением такого недоступного элемента массива был экземпляр агрегата, он удаляется вместе с любыми вложенными в него экземплярами агрегатов. Любые

новые индексные позиции элементов массива, полученные в результате данной команды, будут иметь неустановленные значения. Значение верхнего индекса должно быть большим или равным значению нижнего индекса.

Вход

Агрегат: **array_instance;**
переиндексируемый экземпляр массива.

Нижний: **integer_value;**
значение нового нижнего индекса.

Верхний: **integer_value;**
значение нового верхнего индекса.

Указатели возможных ошибок

MX_NDEF	Доступ к СИДД-модели не определен.
AI_NEXS	Экземпляр агрегата не существует.
AI_NVLD	Экземпляр агрегата не является массивом соответствующей совокупности.
VA_NVLD	Верхний индекс меньше, чем нижний.
VT_NVLD	Значение типа не является целочисленным.
TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
FN_NAVL	Функция не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

10.19 Команды списка прикладных экземпляров

10.19.1 Вставка перед текущим элементом

Данная команда добавляет элемент к экземпляру списка, указанному итератором. Новый элемент вставляется непосредственно перед текущим элементом. При этом позиция текущего элемента не меняется. Если итератор установлен в начале или в конце агрегата, новый элемент становится соответственно первым или последним. Если экземпляр списка пустой, в него вставляется новый элемент, а итератор устанавливается так, как если бы была выполнена команда перехода в конец списка.

Вход

Итератор: **iterator;**
итератор, определяющий экземпляр списка и позицию для вставки.

Значение: **assignable_primitive;**
значение, вставляемое в **Iterator.subject** непосредственно перед текущим элементом.

Указатели возможных ошибок

TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	Доступ к СИДД-модели не имеет типа «чтение—запись».
VT_NVLD	Тип значения неверен.
AI_NEXS	Экземпляр агрегата не существует.
AI_NVLD	Экземпляр агрегата не является списком.
IR_NEXS	Итератор не существует.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

Элемент агрегата **Iterator.subject**, непосредственно предшествующий текущему элементу, указанному Итератором (**Iterator**), должен иметь заданное Значение (**Value**).

Данная команда не должна удалять элемент из **Iterator.subject**.

10.19.2 Вставка после текущего элемента

Данная команда добавляет элемент к экземпляру списка, указанному итератором. Новый элемент вставляется непосредственно после текущего элемента. При этом позиция текущего элемента не меняется. Если итератор установлен в начале или в конце агрегата, новый элемент становится соответственно первым или последним. Если экземпляр списка пустой, в него вставляется новый элемент, а итератор устанавливается так, как если бы была выполнена команда установки в начальное положение.

Вход

Итератор: **iterator;**
итератор, определяющий экземпляр списка и позицию для вставки.

Значение: **assignable_primitive;**
значение, вставляемое в **Iterator.subject** непосредственно после текущего элемента.

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
VT_NVLD Тип значения неверен.
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является списком.
IR_NEXS Итератор не существует.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Элемент агрегата **Iterator.subject**, следующий непосредственно за текущим элементом, указанным Итератором (**Iterator**), должен иметь заданное Значение (**Value**).

Данная команда не должна удалять элемент из **Iterator.subject**.

10.19.3 Вставка по индексу

Данная команда добавляет новый элемент в экземпляр списка. Позиция нового элемента внутри экземпляра списка определяется заданным индексом. В случае, когда значение заданного индекса равно количеству элементов в заданном экземпляре списка плюс один, указанная величина добавляется в конце экземпляра списка. Любое заданное значение индекса, превышающее количество элементов в заданном экземпляре списка плюс один, является неверным.

Вход

Агрегат: **list_instance;**
изменяемый экземпляр списка.

Индекс: **integer_value;**
позиция в списке для нового элемента.

Значение: **assignable_primitive;**
значение, добавляемое в Агрегат (**Aggregate**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является списком.
IX_NVLD Позиция индекса неверна.
VT_NVLD Тип значения неверен.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

В Агрегат (**Aggregate**) на позицию Индекса (**Index**) должно быть добавлено Значение (**Value**).

10.19.4 Создание экземпляра агрегата перед текущим элементом

Данная команда добавляет новый пустой экземпляр агрегата в качестве элемента существующего экземпляра агрегата. Новый агрегат вставляется непосредственно перед позицией, указанной итератором. Если итератор установлен в начале или в конце агрегата, новый экземпляр агрегата становится соответственно первым или последним элементом. Текущий элемент итератора не меняется. Если областью значений экземпляра агрегата, связанного с заданным итератором, является выбираемый тип (SELECT TYPE) языка EXPRESS, **aggregate_primitive**, используемым в качестве ввода/вывода, должен быть **select_aggregate_instance**, а для определения типа (TYPE) экземпляра, создаваемого командой, на входе должен быть установлен атрибут **select_aggregate_instance.data_type**. Если операция требует создания экземпляра массива, который не является

application_indexed_array_instance, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы недостаточна для успешного вычисления выражения, определяющего значение индекса экземпляра массива, должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и реализация не обеспечивает вычисление выражения для индекса массива, должна быть выдана ошибка EX_NSUP.

Вход

Итератор: **iterator;**
итератор Агрегата (**Aggregate**), определяющий позицию, перед которой добавляется значение.

Выход

НовыйАгрегат: **aggregate_primitive;**
новый экземпляр агрегата, добавляемый в **Aggregate** перед позицией, заданной Итератором (**Iterator**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр агрегата не существует.
AI_NVLD Экземпляр агрегата не является списком.
IR_NEXS Итератор не существует.
IR_NSET Текущий элемент итератора не установлен.
VA_NSET Значение индекса экземпляра массива соответствующей совокупности не установлено.
EX_NSUP Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

НовыйАгрегат (**NewAggregate**) должен быть новым пустым экземпляром агрегата, имеющим тип элемента соответствующего Агрегата (**Aggregate**).

NewAggregate должен быть элементом соответствующего **Aggregate**, вставленным в позицию, предшествующую позиции, заданной Итератором (**Iterator**).

10.19.5 Создание экземпляра агрегата после текущего элемента

Данная команда добавляет новый пустой экземпляр агрегата в качестве элемента существующего экземпляра агрегата. Новый экземпляр агрегата добавляется непосредственно после позиции, указанной итератором. Если итератор установлен в начале или в конце агрегата, новый экземпляр агрегата становится соответственно первым или последним элементом. Текущий элемент итератора не меняется. Если областью значений экземпляра агрегата, связанного с заданным итератором, является выбираемый тип (SELECT TYPE) языка EXPRESS, **aggregate_primitive**, используемым в качестве ввода/вывода, должен быть **select_aggregate_instance**, а для определения типа (TYPE) экземпляра, создаваемого командой, на входе должен быть установлен атрибут **select_aggregate_instance.data_type**. Если операция требует создания экземпляра массива, который не является **application_indexed_array_instance**, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы не достаточна для успешного вычисления выражения, определяющего значение индекса экземпляра массива, должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и реализация не обеспечивает вычисление выражения для индекса массива, должна быть выдана ошибка EX_NSUP.

Вход

Итератор: **iterator;**
итератор Агрегата (**Aggregate**), определяющий позицию, после которой добавляется значение.

Выход

НовыйАгрегат: **aggregate_primitive;**
новый экземпляр агрегата, добавляемый в **Aggregate** после позиции, заданной Итератором (**Iterator**).

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».

TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS	Экземпляр агрегата не существует.
AI_NVLD	Экземпляр агрегата не является списком.
IR_NEXS	Итератор не существует.
IR_NSET	Текущий элемент итератора не установлен.
VA_NSET	Значение индекса экземпляра массива соответствующей совокупности не установлено.
EX_NSUP	Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

НовыйАгрегат (**NewAggregate**) должен быть новым пустым экземпляром агрегата, имеющим тип элемента соответствующего Агрегата (**Aggregate**).

NewAggregate должен быть элементом соответствующего **Aggregate**, вставленным в позицию после заданной Итератором (**Iterator**) позиции.

10.19.6 Вставка экземпляра агрегата по индексу

Данная команда добавляет новый пустой экземпляр агрегата в качестве элемента существующего экземпляра агрегата. Позиция нового элемента внутри экземпляра списка определяется заданным индексом. Если значение заданного индекса равно количеству элементов в заданном экземпляре агрегата плюс один, новый экземпляр агрегата добавляется в конце экземпляра списка. Если областью значений заданного экземпляра агрегата является выбираемый тип (SELECT TYPE) языка EXPRESS, **aggregate_primitive**, используемым в качестве ввода/вывода, должен быть **select_aggregate_instance**, а для определения типа (TYPE) экземпляра, создаваемого командой, на входе должен быть установлен атрибут **select_aggregate_instance.data_type**. Если операция требует создания экземпляра массива, который не является **application_indexed_array_instance**, а экземпляр массива не может быть создан, так как существующая совокупность прикладной схемы недостаточна для успешного вычисления выражения, определяющего значение индекса экземпляра массива, должна быть выдана ошибка VA_NSET. Если экземпляр массива не является **application_indexed_array_instance** и реализация не обеспечивает вычисление выражения для индекса массива, должна быть выдана ошибка EX_NSUP.

Вход

Агрегат: **list_instance;**
изменяемый экземпляр списка.

Индекс: **integer_value;**
позиция в списке для нового элемента.

Выход:

НовыйАгрегат: **aggregate_primitive;**
новый экземпляр агрегата, добавляемый в Агрегат (**Aggregate**) в позицию, заданную Индексом (**Index**).

Указатели возможных ошибок

TR_NRW	Транзакция не имеет типа «чтение—запись».
TR_NAVL	Транзакция недоступна в текущем сеансе.
TR_EAB	Транзакция прервана аварийно.
MX_NRW	Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS	Экземпляр списка не существует.
AI_NVLD	Экземпляр агрегата не является списком.
IX_NVLD	Позиция индекса неверна.
VA_NSET	Значение индекса экземпляра массива соответствующей совокупности не установлено.
EX_NSUP	Вычисление выражения индекса не обеспечивается данной реализацией.
SY_ERR	Обнаружена ошибка основной системы.

Влияние на среду СИДД

НовыйАгрегат (**NewAggregate**) должен быть новым пустым экземпляром агрегата, имеющим тип элемента соответствующего Агрегата (**Aggregate**), в позиции Индекса (**Index**).

10.19.7 Удаление по индексу

Данная команда удаляет элемент списка из позиции, установленной заданным индексом. Если элементом списка является экземпляр агрегата, данный экземпляр удаляется вместе со всеми вложенными в него экземплярами агрегатов.

Вход

Агрегат: **list_instance;**
изменяемый экземпляр списка.

Индекс: **integer_value;**
позиция удаляемого элемента в списке.

Указатели возможных ошибок

TR_NRW Транзакция не имеет типа «чтение—запись».
TR_NAVL Транзакция недоступна в текущем сеансе.
TR_EAB Транзакция прервана аварийно.
MX_NRW Доступ к СИДД-модели не имеет типа «чтение—запись».
AI_NEXS Экземпляр списка не существует.
AI_NVLD Экземпляр агрегата не является списком.
IX_NVLD Позиция индекса неверна.
SY_ERR Обнаружена ошибка основной системы.

Влияние на среду СИДД

Должен быть удален элемент Агрегата (**Aggregate**) из позиции, определенной Индексом (**Index**).

Если элемент, удаляемый из **Aggregate**, сам является экземпляром агрегата (**aggregate_instance**), он удаляется вместе со всеми вложенными в него **aggregate_instances**.

11 Ошибки СИДД

В таблице 2 определено множество констант стандартных указателей (индикаторов) ошибок вместе с описанием ошибки и значением кода ошибки, который должен быть установлен в качестве значения атрибута **error_event.error** при обнаружении данной ошибки. Имя константы указателя ошибки формируется комбинацией адресата с предикатом, разделяемых символом подчеркивания. Например, MO_NOPN для ошибки «СИДД-модель не открыта». Для адресатов, приведенных в таблице 2 и определяющих указатели возможных ошибок каждой команды из раздела 10, используются следующие сокращения:

AI — экземпляр агрегата (aggregate instance);
AT — атрибут (attribute);
ED — описание объекта (entity definition);
EI — экземпляр объекта (entity instance);
ER — описание события (event recording);
EX — выражение (expression);
FN — функция (function);
IR — итератор (iterator);
IX — индекс (index);
OP — оператор (operator);
RP — хранилище (repository);
RU — правило (rule);
SC — область действия (scope);
SD — описание схемы (schema definition);
SI — экземпляр схемы (schema instance);
SS — сеанс (session);
MO — СИДД-модель (SDAI-model);
MX — доступ к СИДД-модели (SDAI-model access);
TR — транзакция (transaction);
SY — основная система (underlying system);
VA — значение (value);
VT — тип значения (value type).

Для предикатов, приведенных в таблице 2 и определяющих указатели возможных ошибок каждой команды из раздела 10, используются следующие сокращения:

DUP — дубликат (duplicate);
EAB — прерван аварийно (ended abnormally);
ERR — ошибка (error);
EXS — существует (exists);
NAVL — не доступен (not available);
NDEF — не определен (not defined);
NDEQ — не эквивалентен по области значений (not domain equivalent);
NEXP — неэкспортируемый (not exported);
NEXS — не существует (not exist);
NOPN — не открыт (not open);
NRW — не «чтение—запись» (not read-write);
NSET — не установлен или пустой (not set or empty);
NSUP — не обеспечивается (not supported);
NVLD — неверный (invalid);
OPN — открыт (in open);
RO — «только чтение» (read-only);
RW — «чтение—запись» (read-write).

Состав набора указателей ошибок, который должна обеспечивать реализация СИДД, приведен в таблице 2.

Таблица 2 — Указатели (индикаторы) ошибок СИДД

Указатель ошибки	Описание	Код ошибки	Источник ошибки
SS_OPN	Сеанс открыт	10	**sdai_session**
SS_NAVL	Сеанс не доступен	20	
SS_NOPN	Сеанс не открыт	30	
RP_NEXS	Хранилище не существует	40	
RP_NAVL	Хранилище недоступно	50	**sdai_repository**
RP_OPN	Хранилище открыто	60	**sdai_repository**
RP_NOPN	Хранилище не открыто	70	**sdai_repository**
TR_EAB	Транзакция прервана аварийно	80	**sdai_transaction**
TR_EXS	Транзакция существует	90	**sdai_transaction**
TR_NAVL	Транзакция недоступна в текущем сеансе	100	**sdai_transaction**
TR_RW	Транзакция «чтение—запись»	110	**sdai_transaction**
TR_NRW	Транзакция не «чтение—запись»	120	**sdai_transaction**
TR_NEXS	Транзакция не существует	130	
MO_NDEQ	СИДД-модель не эквивалентна по области значений	140	**sdai_model**
MO_NEXS	СИДД-модель не существует	150	
MO_NVLD	СИДД-модель неверна	160	**sdai_model**
MO_DUP	Дубликат СИДД-модели	170	**sdai_model**
MX_NRW	Доступ к СИДД-модели не имеет типа «чтение—запись»	180	**sdai_model**
MX_NDEF	Доступ к СИДД-модели не определен	190	**sdai_model**
MX_RW	Доступ к СИДД-модели имеет тип «чтение—запись»	200	**sdai_model**
MX_RO	Доступ к СИДД-модели имеет тип «только чтение»	210	**sdai_model**
SD_NDEF	Описание схемы не определено	220	
ED_NDEF	Описание объекта не определено	230	

Окончание таблицы 2

Указатель ошибки	Описание	Код ошибки	Источник ошибки
ED_NDEQ	Описание объекта неэквивалентно по области значений	240	
ED_NVLD	Описание объекта неверно	250	**entity_definition**
RU_NDEF	Правило не определено	260	
EX_NSUP	Вычисление выражения не обеспечивается	270	**bound**
AT_NVLD	Атрибут неверен	280	**attribute**
AT_NDEF	Атрибут не определен	290	
SI_DUP	Дубликат экземпляра схемы	300	**schema_instance**
SI_NEXS	Экземпляр схемы не существует	310	
EI_NEXS	Экземпляр объекта не существует	320	
EI_NAVL	Экземпляр объекта недоступен	330	**application_instance**
EI_NVLD	Экземпляр объекта неверен	340	**entity_instance**
EI_NEXP	Экземпляр объекта не экспортирован	350	**application_instance**
SC_NEXS	Область действия не существует	360	
SC_EXS	Область действия существует	370	
AI_NEXS	Экземпляр агрегата не существует	380	
AI_NVLD	Экземпляр агрегата неверен	390	**aggregate_instance**
AI_NSET	Экземпляр агрегата пустой	400	**aggregate_instance**
VA_NVLD	Значение неверно	410	
VA_NEXS	Значение не существует	420	
VA_NSET	Значение не установлено	430	
VT_NVLD	Тип значение неверен	440	
IR_NEXS	Итератор не существует	450	
IR_NSET	Текущий элемент не определен	460	**iterator**
IX_NVLD	Индекс неверен	470	**aggregate_instance**
ER_NSET	Описание события не установлено	480	
OP_NVLD	Оператор неверен	490	
FN_NAVL	Функция недоступна	500	
SY_ERR	Ошибка основной системы	1000	

12 Модель состояния СИДД

Модели описывают некоторые переходы от одного состояния к другому, происходящие в результате правильного выполнения команды СИДД. Состояние, в соответствии с данным разделом, описывается командами СИДД, допустимыми в определенный момент времени сеанса СИДД. При этом описывается последовательность команд, необходимых для решения определенной задачи во время сеанса СИДД. Правильное выполнение одной команды СИДД может менять или не менять набор допустимых и доступных команд. Например, до выполнения команды «открытие сеанса» допустимой и доступной является только данная команда. После правильного выполнения команды «открытие сеанса» допустимой и доступной станет команда «завершение сеанса». Следовательно, команда «открытие сеанса» изменила состояние сеанса СИДД.

В данном разделе описаны три модели состояния СИДД. Каждая их них основана на конкретном уровне обеспечения транзакции, описанном в 13.1.1. Каждое состояние в каждой модели состояния СИДД имеет определенный набор доступных команд. В таблице 3 команды СИДД сгруппированы по категориям. Данные категории далее используются при описании команд, доступных в каждом состоянии.

Примечание — В данном разделе не определены состояния, зависящие от существования экземпляров объекта и агрегата, областей действия и итераторов. Доступность команд в данных состояниях зависит от наличия входных параметров требуемого экземпляра и допустимости доступа к заданному экземпляру в режиме «только чтение» или «чтение—запись».

Таблица 3 — Группирование команд СИДД

Категория	Описание	Команда
A	Инициирование	10.3.1 Открытие сеанса
B	Уровень 3 Старт	10.4.6 Начало транзакции с доступом «чтение—запись» 10.4.7 Начало транзакции с доступом «только чтение»
C	Сеанс	10.4.1 Запись ошибки 10.4.2 Начало описания события 10.4.3 Окончание описания события 10.4.4 Закрытие сеанса 10.4.12 Создание нефиксированного списка 10.4.13 Удаление нефиксированного списка 10.4.14 Запрос СИДД
D	Открытие хранилища	10.4.5 Открытие хранилища
E	Уровень 3 Продолжение существования	10.4.8 Фиксация транзакции 10.4.9 Аварийное прерывание 10.4.10 Завершение доступа и фиксации транзакции 10.4.11 Завершение доступа к транзакции и аварийное прерывание
F	Закрытие хранилища	10.5.3 Закрытие хранилища
G	Чтение модели	10.7.3 Начало доступа «только чтение»
H	Чтение экземпляра схемы	10.6.5 Проверка глобального правила 10.6.6 Проверка правила уникальности 10.6.7 Проверка области значений ссылки на экземпляр 10.6.9 Определение актуальности проверки
I	Создание объектов	10.5.1 Создание СИДД-модели 10.5.2 Создание экземпляра схемы
J	Запись модели	10.7.6 Начало доступа «чтение—запись» 10.7.1 Удаление СИДД-модели 10.7.2 Переименование СИДД-модели
K	Запись экземпляра схемы	10.6.3 Добавление СИДД-модели 10.6.4 Удаление СИДД-модели 10.6.1 Удаление экземпляра схемы 10.6.2 Переименование экземпляра схемы 10.6.8 Проверка экземпляра схемы
L	Модель словаря	10.9.1 Получение определения сложного объекта 10.9.2 Проверка принадлежности к подтипу 10.9.3 Проверка принадлежности к подтипу СИДД 10.9.4 Проверка эквивалентности областей значений

Продолжение таблицы 3

Категория	Описание	Команда
M	Чтение экземпляра объекта	10.7.8 Получение определения объекта 10.10.1 Получение значения атрибута 10.10.2 Проверка атрибута 10.10.3 Поиск СИДД-модели экземпляра объекта 10.10.4 Получение типа экземпляра 10.10.5 Определение соответствия экземпляра заданному типу 10.10.6 Определение соответствия экземпляра типу прикладной схемы 10.10.7 Определение соответствия экземпляра типу прикладной схемы и схемы параметризованных данных СИДД 10.10.8 Поиск пользователей экземпляра объекта 10.10.9 Поиск пользователей экземпляра объекта в заданной роли 10.10.10 Получение значения границы атрибута 10.10.11 Поиск ролей экземпляра 10.10.12 Поиск типов данных экземпляра 10.11.6 Получение постоянной метки 10.11.7 Получение идентификатора сеанса 10.11.8 Получение описания 10.11.9 Проверка правила «were» 10.11.10 Проверка наличия значений у явных атрибутов 10.11.11 Проверка инверсных атрибутов 10.11.12 Проверка ссылок явных атрибутов 10.11.16 Проверка ширины строки 10.11.17 Проверка ширины двоичного значения 10.11.18 Проверка точности действительного значения
N	Чтение агрегата	10.11.13 Проверка размерности агрегатов 10.11.14 Проверка уникальности агрегатов 10.11.15 Проверка массива на наличие пустых элементов 10.12.1 Получение количества элементов 10.12.2 Проверка на вхождение в экземпляр агрегата 10.12.3 Создание итератора 10.12.9 Получение нижней границы 10.12.10 Получение верхней границы 10.15.1 Получение по индексу 10.15.4 Получение значения границы по индексу 10.17.1 Проверка наличия значения по индексу 10.17.3 Получение нижнего индекса 10.17.4 Получение верхнего индекса
O	Чтение итератора	10.12.4 Удаление итератора 10.12.5 Установка в начальное положение 10.12.6 Переход к новому текущему элементу 10.12.7 Получение текущего элемента 10.12.8 Получение значения границы по итератору 10.15.2 Переход в конец 10.15.3 Переход к предыдущему элементу 10.17.2 Проверка текущего элемента
P	Чтение области действия	10.8.2 Определение владельца области действия 10.8.3 Получение области действия 10.8.9 Проверка ссылочных ограничений области действия
Q	Завершение доступа к модели в режиме «только чтение»	10.7.5 Завершение доступа «только чтение»
R	Перевод модели	10.7.4 Перевод СИДД-модели в режим «чтение—запись»

Окончание таблицы 3

Категория	Описание	Команда
S	Уровень 2 Продолжение существования	10.7.10 Отмена изменений 10.7.11 Сохранение изменений
T	Запись модели	10.7.7 Завершение доступа «чтение—запись» 10.7.9 Создание экземпляра объекта
U	Запись прикладного экземпляра	10.11.1 Копирование прикладного экземпляра 10.11.2 Удаление прикладного экземпляра 10.11.3 Установка значения атрибута 10.11.4 Возврат атрибута в неустановленное значение 10.11.5 Создание экземпляра агрегата
V	Запись экземпляра агрегата	10.14.1 Неупорядоченное добавление 10.14.2 Неупорядоченное создание экземпляра агрегата 10.14.3 Неупорядоченное удаление 10.16.1 Внесение значения по индексу 10.16.2 Создание экземпляра агрегата по индексу 10.18.1 Возврат к неустановленному значению по индексу 10.18.3 Переиндексирование массива 10.18.4 Переустановка индексов массива 10.19.3 Вставка по индексу 10.19.6 Вставка экземпляра агрегата по индексу 10.19.7 Удаление по индексу
W	Запись итератора	10.13.1 Создание экземпляра агрегата в качестве текущего элемента 10.13.2 Задание значения текущему элементу 10.13.3 Удаление текущего элемента 10.18.2 Возврат текущего элемента к неустановленному значению 10.19.1 Вставка перед текущим элементом 10.19.2 Вставка после текущего элемента 10.19.4 Создание экземпляра агрегата перед текущим элементом 10.19.5 Создание экземпляра агрегата после текущего элемента
X	Запись области действия	10.8.1 Пополнение области действия 10.8.4 Удаление из области действия 10.8.5 Добавление к экспортному списку 10.8.6 Удаление из экспортного списка 10.8.7 Удаление области действия 10.8.8 Копирование области действия

12.1 Модель состояния для транзакции уровня 1

Модель содержит пять состояний. Эти состояния вместе с возможными переходами между ними определены в 12.1.1—12.1.6. Команды в категориях B, E и S не обязаны обеспечиваться реализациями транзакции уровня 1.

12.1.1 Состояние «Нет сеанса 1» (No Session 1)

Устанавливается перед вызовом команды «открытие сеанса». В этом состоянии доступны команды категории A. Команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.1.2 Состояние «Сеанс 1» (Session 1)

Устанавливается после вызова команды «открытие сеанса». В этом состоянии доступны команды категорий C и D. Команды категорий A, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.1.3 Состояние «Открытое хранилище 1» (Repository Open 1)

Устанавливается сразу после вызова в состояние «Сеанс 1» команды «открытие хранилища». В этом состоянии доступны команды категорий C, D, F, G, H, I, J и K. Команды категорий A, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.1.4 Состояние «Начало доступа к СИДД-модели в режиме «только чтение» 1» (SDAI-model Started RO 1)

Устанавливается сразу после вызова в состояние «Открытое хранилище 1» команды «начало доступа «только чтение»». В этом состоянии доступны команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, Q и R. Команды категорий A, T, U, V, W и X в этом состоянии недоступны.

12.1.5 Состояние «Начало доступа к СИДД-модели в режиме «чтение—запись» 1» (SDAI-model Started RW 1)

Устанавливается сразу после вызова в состояние «Открытое хранилище 1» команды «начало доступа «чтение—запись»». В этом состоянии доступны команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, T, U, V, W и X. Команды категорий A, Q и R в этом состоянии недоступны.

12.1.6 Переходы состояний

Переходы из одного состояния в другое определены в таблице 4. В таблице строки 1—5 представляют исходное состояние, графы 2—6 — результирующее состояние, а значения, находящиеся на пересечении строки с графой, — команды перехода от начального состояния в результирующее. Символ «x» обозначает случай, когда строка и графа представляют одно и то же состояние. Символ «—» обозначает случай, когда не существует единственной команды перехода между данными состояниями.

Таблица 4 — Переходы состояний для транзакции уровня 1

Состояние	Нет сеанса 1	Сеанс 1	Открытое хранилище 1	Начало доступа к СИДД-модели в режиме «только чтение» 1	Начало доступа к СИДД-модели в режиме «чтение—запись» 1
1	2	3	4	5	6
Нет сеанса 1	x	10.3.1	—	—	—
Сеанс 1	10.4.4	x	10.4.5	—	—
Открытое хранилище 1	10.4.4	10.5.3	x	10.7.3	10.7.6
Начало доступа к СИДД-модели в режиме «только чтение» 1	10.4.4	10.5.3	10.7.5 10.7.1	x	10.7.4
Начало доступа к СИДД-модели в режиме «чтение—запись» 1	10.4.4	10.5.3	10.7.7 10.7.1	—	x

12.2 Модель состояния для транзакции уровня 2

Модель содержит пять состояний. Эти состояния вместе с возможными переходами между ними определены в 12.2.1—12.2.6. Команды категорий B и E не обязаны обеспечиваться реализациями транзакции уровня 2.

12.2.1 Состояние «Нет сеанса 2» (No Session 2)

Устанавливается перед вызовом команды «открытие сеанса». В этом состоянии доступны команды категории A. Команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.2.2 Состояние «Сеанс 2» (Session 2)

Устанавливается после вызова команды «открытие сеанса». В этом состоянии доступны команды категорий C и D. Команды категорий A, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W и X в этом состоянии недоступны.

12.2.3 Состояние «Открытое хранилище 2» (Repository Open 2)

Устанавливается сразу после вызова в состояние «Сеанс 2» команды «открытие хранилища». В этом состоянии доступны команды категорий C, D, F, G, H, I, J и K. Команды категорий A, L, M, N, O, P, Q, R, S, T, U, V, W и X в этом состоянии недоступны.

12.2.4 Состояние «Начало доступа к СИДД-модели в режиме «только чтение» 2» (SDAI-model Started RO 2)

Устанавливается сразу после вызова в состояние «Открытое хранилище 2» команды «начало доступа «только чтение». В этом состоянии доступны команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, Q и R. Команды категорий A, S, T, U, V, W и X в этом состоянии недоступны.

12.2.5 Состояние «Начало доступа к СИДД-модели в режиме «чтение—запись» 2» (SDAI-model Started RW 2)

Устанавливается сразу после вызова в состояние «Открытое хранилище 2» команды «начало доступа «чтение—запись». В этом состоянии доступны команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, S, T, U, V, W и X. Команды категорий A, Q и R в этом состоянии недоступны.

12.2.6 Переходы состояний

Переходы из одного состояния в другое определены в таблице 5. В таблице строки 1—5 представляют исходное состояние, графы 2—6 — результирующее состояние, а значения, находящиеся на пересечении строки с графой — команды перехода из исходного состояния в результирующее. Символ «x» обозначает случай, когда строка и графа представляют одно и то же состояние. Символ «—» обозначает случай, когда не существует единственной команды перехода между данными состояниями.

Таблица 5 — Переходы состояний для транзакции уровня 2

Состояние	Нет сеанса 2	Сеанс 2	Открытое хранилище 2	Начало доступа к СИДД-модели в режиме «только чтение» 2	Начало доступа к СИДД-модели в режиме «чтение—запись» 2
1	2	3	4	5	6
Нет сеанса 2	x	10.3.1	—	—	—
Сеанс 2	10.4.4	x	10.4.5	—	—
Открытое хранилище 2	10.4.4	10.5.3	x	10.7.3	10.7.6
Начало доступа к СИДД-модели в режиме «только чтение» 2	10.4.4	10.5.3	10.7.5 10.7.1	x	10.7.4
Начало доступа к СИДД-модели в режиме «чтение—запись» 2	10.4.4	10.5.3	10.7.7 10.7.1	—	x

12.3 Модель состояния для транзакции уровня 3

Модель содержит десять состояний. Эти состояния вместе с возможными переходами между ними определены в 12.3.1—12.3.11. Команды категории S не обязаны обеспечиваться реализациями транзакции уровня 3.

12.3.1 Состояние «Нет сеанса 3» (No Session 3)

Устанавливается перед вызовом команды «открытие сеанса». В этом состоянии доступны команды категории A. Команды категорий C, D, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.2 Состояние «Сеанс 3» (Session 3)

Устанавливается после вызова команды «открытие сеанса». В этом состоянии доступны команды категорий B и C. Команды категорий A, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.3 Состояние «Начало транзакции в режиме «только чтение» 3» (Transaction Started RO 3)

Устанавливается сразу после вызова в состояние «Сеанс 3» команды «начало транзакции с доступом «только чтение». В этом состоянии доступны команды категорий C, D и E. Команды категорий A, B, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.4 Состояние «Начало транзакции в режиме «чтение—запись» 3» (Transaction Started RW 3)

Устанавливается сразу после вызова в состояние «Сеанс 3» команды «начало транзакции с доступом «чтение—запись». В этом состоянии доступны команды категорий C, D и E. Команды категорий A, B, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.5 Состояние «Открытое хранилище 3» (Repository Open 3)

Устанавливается сразу после вызова в состояние «Сеанс 3» команды «открытие хранилища». В этом состоянии доступны команды категорий B, C, D и F. Команды категорий A, E, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.6 Состояние «Открытое хранилище в режиме «только чтение» 3» (RO Repository Open 3)

Устанавливается сразу после вызова в состояние «Начало транзакции в режиме «только чтение» 3» команды «открытие хранилища» или после вызова в состояние «Открытое хранилище 3» команды «начало транзакции с доступом «только чтение». В этом состоянии доступны команды категорий C, D, E, F, G и H. Команды категорий A, B, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.7 Состояние «Открытое хранилище в режиме «чтение—запись» 3» (RW Repository Open 3)

Устанавливается сразу после вызова в состояние «Начало транзакции «чтение—запись» 3» команды «открытие хранилища» или после вызова в состояние «Открытое хранилище 3» команды «начало транзакции с доступом «чтение—запись». В этом состоянии доступны команды категорий C, D, E, F, G, H, I, J и K. Команды категорий A, B, L, M, N, O, P, Q, R, T, U, V, W и X в этом состоянии недоступны.

12.3.8 Состояние «Начало доступа к модели в режиме «только чтение» в хранилище с доступом «только чтение» 3» (RO Model Started RO 3)

Устанавливается сразу после вызова в состояние «Открытое хранилище в режиме «только чтение» 3» команды «начало доступа «только чтение»». В этом состоянии доступны команды категорий C, D, E, F, G, H, L, M, N, O, P и Q. Команды категорий A, B, I, J, K, R, T, U, V, W и X в этом состоянии недоступны.

12.3.9 Состояние «Начало доступа к модели в режиме «только чтение» в хранилище с доступом «чтение—запись» 3» (RW Model Started RO 3)

Устанавливается сразу после вызова в состояние «Открытое хранилище в режиме «чтение—запись» 3» команды «начало доступа «только чтение». В этом состоянии доступны команды категорий C, D, E, F, G, H, I, K, L, M, N, O, P, Q и R. Команды категорий A, B, J, T, U, V, W и X в этом состоянии недоступны.

12.3.10 Состояние «Начало доступа к модели в режиме «чтение—запись» в хранилище с доступом «чтение—запись» 3» (RW Model Started RW 3)

Устанавливается сразу после вызова в состояние «Открытое хранилище в режиме «чтение—запись» 3» команды «начало доступа «чтение—запись». В этом состоянии доступны команды категорий C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, T, U, V, W и X. Команды категорий A и B в этом состоянии недоступны.

12.3.11 Переходы состояний

Переходы из одного состояния в другое определены в таблице 6. В таблице строки 1—10 представляют исходное состояние, графы 2—11 — результирующее состояние, а значения, находящиеся на пересечении строки с графой, — команды перехода из исходного состояния в результирующее. Символ «x» обозначает случай, когда строка и графа представляют одно и то же состояние. Символ «—» обозначает случай, когда не существует единственной команды перехода между данными состояниями.

В таблице 6 использованы следующие сокращения.

N — состояние «Нет сеанса 3»;

S — состояние «Сеанс 3»;

R — состояние «Начало транзакции в режиме «только на чтение» 3»;

W — состояние «Начало транзакции в режиме «чтение—запись» 3»;
RO — состояние Открытое хранилище 3»;
RP — состояние «Открытое хранилище в режиме «только чтение» 3»;
WP — состояние «Открытое хранилище в режиме «чтение—запись» 3»;
RR — состояние «Начало доступа к модели в режиме «только чтение» в хранилище с доступом «только чтение» 3»;
WR — состояние «Начало доступа к модели в режиме «только чтение» в хранилище с доступом «чтение—запись» 3»;
WW — состояние «Начало доступа к модели в режиме «чтение—запись» в хранилище с доступом «чтение—запись» 3».

Таблица 6 — Переходы состояний для транзакций уровня 3

Состояние	N	S	R	W	RO	RP	WP	RR	WR	WW
N	x	10.3.1	—	—	—	—	—	—	—	—
S	10.4.4	x	10.4.7	10.4.6	10.4.5	—	—	—	—	—
R	10.4.4	10.4.10 10.4.11	x	—	—	10.4.5	—	—	—	—
W	10.4.4	10.4.10 10.4.11	—	x	—	—	10.4.5	—	—	—
RO	10.4.4	10.5.3	—	—	x	10.4.7	10.4.6	—	—	—
RP	10.4.4	—	10.5.3	—	10.4.10 10.4.11	x	—	10.7.3	—	—
WP	10.4.4	—	—	10.5.3	10.4.10 10.4.11	—	x	—	10.7.3	10.7.6
RR	10.4.4	—	10.5.3	—	—	10.7.5	—	x	—	—
WR	10.4.4	—	—	10.5.3	—	—	10.7.5 10.7.1	—	x	10.7.4
WW	10.4.4	—	—	10.5.3	—	—	10.7.7 10.7.1	—	—	x

13 Классы реализации

13.1 Реализации СИДД

Реализация, соответствующая одной или нескольким языковым привязкам СИДД, должна быть совместима с заданным уровнем функциональных возможностей, установленным ниже. Заявка о соответствии реализации протоколу (ЗСРП, PICS) для конкретной реализации СИДД должна устанавливать уровни, которым она удовлетворяет. Когда данной реализацией не обеспечивается команда или параметры, с которыми она вызывается, реализация должна предусматривать данную команду, но в результате ее вызова должна возвращать ошибку FN_NAVL.

13.1.1 Уровни транзакции

В настоящем стандарте установлены три уровня реализации транзакции:

1 — **нет транзакций.** Уровень состоит из реализации, не предусматривающей обеспечение команд на получение постоянной метки, транзакции сеанса и сохранение и отмену изменений СИДД-модели;

2— **транзакции СИДД-моделей.** Уровень состоит из реализации, предусматривающей обеспечение команд на получение постоянной метки и сохранение и отмену изменений СИДД-модели, но не обеспечивающей команды на транзакцию сеанса;

3 — **транзакции.** Уровень состоит из реализации, предусматривающей обеспечение команд на получение постоянной метки и транзакции сеанса, но не обеспечивающей команды на сохранение и отмену изменений СИДД-модели.

13.1.2 Уровни вычисления выражения для проверки и вычисляемых атрибутов

В настоящем стандарте установлены четыре уровня обеспечения вычисления выражения. Реализация СИДД, обеспечивающая заданный уровень вычисления выражения, должна также обеспечивать все более низкие уровни:

1 — **нет вычисления.** Уровень не предусматривает обеспечения любой команды на проверку, запрос СИДД, получение определения сложного объекта, поиск пользователей экземпляра объекта, получение значения атрибута для вычисляемых и инверсных атрибутов, объявленных в прикладных схемах, или доступ к экземплярам объектов словаря СИДД. Ограничения, установленные в любой схеме СИДД, должны быть выполнены всегда, независимо от наличия команд на проверку;

2 — **простое вычисление.** Уровень содержит обеспечение, предусмотренное на Уровне 1, плюс доступ к экземплярам объектов словаря СИДД, получение значений инверсных атрибутов, объявленных в прикладных схемах, и все команды на проверку, за исключением проверок экземпляра схемы, глобального правила и правила «were», со следующими ограничениями: проверку размерности агрегата, точности действительного значения, ширины строки и ширины двоичного значения необходимо обеспечивать только в случаях, когда в прикладной схеме границы заданы явными целыми числами. Ограничения, установленные в схеме словаря СИДД, должны быть выполнены всегда, независимо от наличия команд на проверку;

3 — **сложное вычисление.** Уровень содержит обеспечение, предусмотренное на Уровне 2, плюс обеспечение запроса СИДД, поиск пользователей экземпляра объекта и следующие команды: на получение значений вычисляемых атрибутов, проверки правила «were», экземпляра схемы, глобального правила, размерности агрегата, точности действительного значения, ширины строки и ширины двоичного значения, за исключением случаев, когда включенное вычисление выражения содержит вложенный запрос, функцию определения прикладной схемы, встроенные функции «USEDIN» или «ROLESOF» языка EXPRESS;

4 — **полное вычисление.** Уровень содержит обеспечение, предусмотренное на Уровне 3, плюс обеспечение полного набора команд на проверки, получение определения сложного объекта и значений для всех атрибутов.

13.1.3 Уровни записи событий сеанса

В настоящем стандарте установлены два уровня записи событий сеанса:

1 — **нет записи.** Уровень не содержит функции обеспечения записи событий сеанса. На этом уровне не создаются экземпляры типа объекта **error_event,** список **sdai_session.errors** всегда пуст, а значение атрибута **session.recording_active** всегда FALSE;

2 — **обеспечение записи.** Уровень предусматривает функцию обеспечения записи событий сеанса. Каждая команда, выдающая ошибку, создает экземпляр **error_event,** и этот экземпляр добавляется в конце списка **sdai_session.errors.**

13.1.4 Уровни области действия

В настоящем стандарте установлены два уровня обеспечения конструкции SCOPE (ГОСТ Р ИСО 10303-21):

1 — **нет области действия.** Уровень не содержит обеспечения команд на область действия;

2 — **обеспечение области действия.** Уровень предусматривает обеспечение команд на область действия.

13.1.5 Уровни эквивалентности области значений

В настоящем стандарте установлены два уровня эквивалентности области значений:

1 — **нет эквивалентности области значений.** Уровень не содержит обеспечение объявления эквивалентности области значений типов объекта и его использование прикладными экземплярами, основанными на различных определениях схем;

2 — **эквивалентность области значений.** Уровень предусматривает обеспечение объявления эквивалентности области значений типов объекта и его использование прикладными экземплярами, основанными на различных определениях схем. Реализация должна обеспечивать один или оба алгоритма заполнения информации об эквивалентности области значений по А.2.2 или специфический для реализации механизм заполнения информации об эквивалентности области значений.

13.2 Спецификация классов реализаций

В данном подразделе определены возможные классы реализаций, на соответствие которым могут быть заявлены реализации СИДД.

13.2.1 Реализация класса 1

Представляет собой самый низкий уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать уровни 1:

- транзакции;
- вычисления выражения;
- запись сеанса;
- область действия;
- эквивалентность области значений.

13.2.2 Реализация класса 2

Представляет собой второй уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать:

- уровень 1 — транзакции;
- уровень 2 — вычисления выражения;
- уровень 1 — запись сеанса;
- уровень 2 — область действия;
- уровень 1 — эквивалентность области значений.

13.2.3 Реализация класса 3

Представляет собой третий уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать:

- уровень 1 — транзакции;
- уровень 3 — вычисления выражения;
- уровень 1 — запись сеанса;
- уровень 2 — область действия;
- уровень 2 — эквивалентность области значений.

13.2.4 Реализация класса 4

Представляет собой четвертый уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать:

- уровень 2 — транзакции;
- уровень 2 — вычисления выражения;
- уровень 2 — запись сеанса;
- уровень 1 — область действия;
- уровень 1 — эквивалентность области значений.

13.2.5 Реализация класса 5

Представляет собой пятый уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать:

- уровень 3 — транзакции;
- уровень 2 — вычисления выражения;
- уровень 2 — запись сеанса;
- уровень 1 — область действия;
- уровень 1 — эквивалентность области значений.

13.2.6 Реализация класса 6

Представляет собой шестой уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать:

- уровень 3 — транзакции;
- уровень 3 — вычисления выражения;
- уровень 2 — запись сеанса;
- уровень 1 — область действия;
- уровень 2 — эквивалентность области значений.

13.2.7 Реализация класса 7

Представляет собой седьмой уровень функций, которому может соответствовать любая реализация СИДД. Реализации данного класса должны обеспечивать:

- уровень 3 — транзакции;
- уровень 4 — вычисления выражения;
- уровень 2 — запись сеанса;
- уровень 2 — область действия;
- уровень 2 — эквивалентность области значений.

13.3 Команды, необходимые для классов реализаций

В таблице 7 описаны команды, необходимые для каждого класса реализации. В строках таблицы указаны команды, а в графах — классы реализации 1—7. Символ «Y» обозначает, что необходимо обеспечение полных функциональных возможностей данной команды, «N» — от команды требуется только возврат указателя ошибки FN_NAVL, «D» — необходимо обеспечение функциональных возможностей команды, за исключением обеспечения эквивалентности области значений, а «E» — необходимо обеспечение функциональных возможностей данной команды, соответствующих уровню оценки выражения, установленному для данного класса реализации.

Таблица 7 — Команды, необходимые для класса реализации

Команда	Класс реализации						
	1	2	3	4	5	6	7
10.3.1 Открытие сеанса	Y	Y	Y	Y	Y	Y	Y
10.4.1 Запись ошибки	N	N	N	Y	Y	Y	Y
10.4.2 Начало описания события	N	N	N	Y	Y	Y	Y
10.4.3 Окончание описания события	N	N	N	Y	Y	Y	Y
10.4.4 Закрытие сеанса	Y	Y	Y	Y	Y	Y	Y
10.4.5 Открытие хранилища	Y	Y	Y	Y	Y	Y	Y
10.4.6 Начало транзакции с доступом «чтение—запись»	N	N	N	N	Y	Y	Y
10.4.7 Начало транзакции с доступом «только чтение»	N	N	N	N	Y	Y	Y
10.4.8 Фиксация транзакции	N	N	N	N	Y	Y	Y
10.4.9 Аварийное прерывание	N	N	N	N	Y	Y	Y
10.4.10 Завершение доступа и фиксации транзакции	N	N	N	N	Y	Y	Y
10.4.11 Завершение доступа к транзакции и аварийное прерывание	N	N	N	N	Y	Y	Y
10.4.12 Создание нефиксированного списка	Y	Y	Y	Y	Y	Y	Y
10.4.13 Удаление нефиксированного списка	Y	Y	Y	Y	Y	Y	Y
10.4.14 Запрос СИДД	N	N	Y	N	N	Y	Y
10.5.1 Создание СИДД-модели	Y	Y	Y	Y	Y	Y	Y
10.5.2 Создание экземпляра схемы	Y	Y	Y	Y	Y	Y	Y
10.5.3 Закрытие хранилища	Y	Y	Y	Y	Y	Y	Y
10.6.1 Удаление экземпляра схемы	Y	Y	Y	Y	Y	Y	Y
10.6.2 Переименование экземпляра схемы	Y	Y	Y	Y	Y	Y	Y
10.6.3 Добавление СИДД-модели	D	D	Y	D	D	Y	Y
10.6.4 Удаление СИДД-модели	Y	Y	Y	Y	Y	Y	Y
10.6.5 Проверка глобального правила	N	N	E	N	N	E	Y
10.6.6 Проверка правила уникальности	N	Y	Y	Y	Y	Y	Y
10.6.7 Проверка области значений ссылки на экземпляр	N	Y	Y	Y	Y	Y	Y
10.6.8 Проверка экземпляра схемы	N	Y	Y	Y	Y	Y	Y
10.6.9 Определение актуальности проверки	N	Y	Y	Y	Y	Y	Y
10.7.1—10.7.7 Команды СИДД-модели, кроме команд на сохранение или отмену изменений, получение определения объекта и создание экземпляра объекта	Y	Y	Y	Y	Y	Y	Y
10.7.8 Получение определения объекта	N	Y	Y	Y	Y	Y	Y

Окончание таблицы 7

Команда	Класс реализации						
	1	2	3	4	5	6	7
10.7.9 Создание экземпляра объекта	Y	Y	Y	Y	Y	Y	Y
10.7.10, 10.7.11 Отмена и сохранение изменений	N	N	N	Y	N	N	N
10.8.1—10.8.9 Область действия	N	Y	Y	N	N	N	Y
10.9.1 Получение определения сложного объекта	N	N	N	N	N	N	Y
10.9.2 Проверка принадлежности к подтипу	Y	Y	Y	Y	Y	Y	Y
10.9.3 Проверка принадлежности к подтипу СИДД	Y	Y	Y	Y	Y	Y	Y
10.9.4 Проверка эквивалентности областей значений	N	N	Y	N	Y	Y	Y
10.10.1 Получение значения атрибута	E	E	E	E	E	E	Y
10.10.4 Получение типа экземпляра	N	Y	Y	Y	Y	Y	Y
10.10.2, 10.10.3, 10.10.5—10.10.7 Экземпляров объектов, кроме команд на получение значения атрибута или получение типа экземпляра	Y	Y	Y	Y	Y	Y	Y
10.10.8 Поиск пользователей экземпляра объекта	N	N	Y	N	N	Y	Y
10.10.9 Поиск пользователей экземпляра объекта в заданной роли	N	N	Y	N	N	Y	Y
10.10.10 Получение значения границы атрибута	E	E	E	E	E	E	Y
10.10.11 Поиск ролей экземпляра	N	N	N	N	N	N	Y
10.10.12 Поиск типов данных экземпляра	N	N	N	N	N	N	Y
10.11.1 Копирование прикладного экземпляра	D	D	Y	D	D	Y	Y
10.11.2—10.11.5 Прикладного экземпляра, кроме команд на проверки и метки	Y	Y	Y	Y	Y	Y	Y
10.11.6, 10.11.7 Прикладной экземпляр — команда постоянной метки	N	N	N	Y	Y	Y	Y
10.11.8 Получение описания	Y	Y	Y	Y	Y	Y	Y
10.11.10—10.11.12 Прикладной экземпляр — простая проверка	N	Y	Y	Y	Y	Y	Y
10.11.9, 10.11.13—10.11.18 Прикладной экземпляр — проверки	N	E	E	E	E	E	Y
10.12.8—10.12.10, 10.15.4, 10.17.3, 10.17.4, 10.18.3, 10.18.4 Агрегаты, связанные с границами объекта и прикладного экземпляра	N	E	E	E	E	E	Y
10.12.1—10.12.7, 10.13.1—10.15.3, 10.16.1—10.17.2, 10.18.1, 10.18.2, 10.19.1—10.19.7 Агрегаты, не связанные с границами какого-либо объекта и прикладного экземпляра	Y	Y	Y	Y	Y	Y	Y

ПРИЛОЖЕНИЕ А
(обязательное)

Отображение конструкций языка EXPRESS в конструкции схемы словаря СИДД

А.1 Конструкции языка EXPRESS

Схема словаря СИДД не обеспечивает напрямую все конструкции языка EXPRESS. Не обеспечиваются конструкции, не имеющие отношения к командам СИДД. В настоящем приложении описаны преобразования, которые должны быть выполнены со схемой при ее отображении в представление схемы словаря СИДД, в дальнейшем называемой словарем данных.

Для каждой схемы, доступной приложению посредством реализации СИДД как части словаря данных, должна существовать СИДД-модель, содержащая соответствующую информацию словаря (см. 6.1). Значением атрибута **sdai_model.name** каждой СИДД-модели должно быть имя схемы, выделенное прописными буквами (символами) с добавлением окончания «_DICTIONARY_DATA». Для экземпляра схемы, с которым связаны все СИДД-модели — части словаря данных, значением атрибута **schema_instance.name** должно быть «SDAI_DICTIONARY_SCHEMA_INSTANCE», а каждая СИДД-модель должна быть элементом множества **schema_instance.associated_models** (см. 8.4.1). Экземпляр схемы и связанные с ним СИДД-модели должны быть основаны на схеме сеанса СИДД.

А.1.1 Спецификация интерфейса

Когда словарь данных заполняется на основе конкретной схемы, все элементы языка EXPRESS, явно или неявно импортированные в текущую схему, должны быть допустимы в ней. Все импортированные элементы рассматриваются как часть экземпляра **schema_definition**, представляющего текущую схему. Схемы, из которых импортируются элементы, не должны быть экземплярами **schema_definition** до тех пор, пока они не будут обработаны индивидуально.

Все описания, ставшие явно или неявно видимыми в текущей схеме посредством операторов USE или REFERENCE, должны быть представлены в словаре. Когда тип данных объекта стал видимым при помощи REFERENCE или был неявно импортирован, или в обоих случаях, его словарный компонент должен иметь атрибут **entity_definition.independent** со значением FALSE. Все другие определения объекта должны иметь этот атрибут со значением TRUE.

Конструкции в схеме словаря СИДД, обеспечивающие команды, определенные в настоящем стандарте, не учитывают спецификацию интерфейса. Однако для приложений, могущих использовать спецификации интерфейса, схема предусматривает конструкции, обеспечивающие доступ приложения к данной информации. Данные конструкции запоминают тот факт, что эти элементы словаря были импортированы в текущий экземпляр **schema_definition**, но в настоящем стандарте не предусмотрены команды или функция, могущие воспользоваться этой информацией (см. 6.4.2—6.4.7).

А.1.2 Интерпретация конструкции ABSTRACT языка EXPRESS

Если тип объекта объявлен абстрактным (ABSTRACT) супертипом, значением его атрибута **entity_definition.instantiable** должно быть FALSE, в противном случае — TRUE.

А.1.3 Интерпретация выражений AND и ANDOR в конструкции SUPERTYPE языка EXPRESS

Если объявление объекта (ENTITY) языка EXPRESS является супертипом, оператор SUPERTYPE которого содержит одно или несколько выражений AND или ANDOR, или если многочисленные объявления ENTITY языка EXPRESS определены как подтипы одного супертипа, а супертип не задает ограничение ONEOF, приложение В ГОСТ Р ИСО 10303—11 устанавливает, как найти счетное множество определений типа данных объекта, для которых потенциально можно создать экземпляры. Существуют два способа, по которым эти типы данных объекта могут быть сделаны доступными посредством реализации СИДД:

- реализация СИДД может обеспечивать окончательно заполненный словарь СИДД для экземпляров **entity_definition** с полным множеством определений типа данных объекта, делая их после команды «начало доступа «только чтение» доступными для СИДД-модели, содержащей информацию словаря данных для конкретной схемы;

- реализация СИДД может обеспечивать создание экземпляров **entity_definition**, основанных на определениях типа данных объекта, полученных в результате выполнения команды «получение определения сложного объекта».

Внутри конкретного сложного экземпляра объекта С определяется корневой объект как не имеющий в С экземпляров подтипов. Для каждого допустимого сложного экземпляра, создаваемого по алгоритму, определенному в приложении В ГОСТ Р ИСО 10303-11, и содержащего несколько корневых объектов, должен быть создан дополнительный экземпляр **entity_definition**, который должен быть подтипом каждого из корневых объектов в сложном экземпляре. Значение атрибута **entity_definition.name** должно формироваться путем соединения имен типов корневых объектов в алфавитном порядке, разделенных символом плюс (+).

Пример 17 — Предположим, что имеются следующие EXPRESS-схемы:

```
SCHEMA resource;
ENTITY a ABSTRACT SUPERTYPE OF (b ANDOR c);
    attr1 : REAL;
WHERE
    WR1 : attr1 >= 0.0;
END_ENTITY;
ENTITY b SUBTYPE OF (a); END_ENTITY;
ENTITY c SUBTYPE OF (a);
    attr4 : STRING;
UNIQUE
    UR1 : attr4;
END_ENTITY;
END_SCHEMA;
SCHEMA example_schema;
USE FROM resource (a, b);
REFERENCE FROM resource (c);
ENTITY d SUPERTYPE OF (ONEOF (e, f) AND (g ANDOR h)) SUBTYPE OF (b);
    att2 : g;
END_ENTITY;
ENTITY e SUBTYPE OF (d);
    SELF\a.attr1 : INTEGER;
END_ENTITY;
ENTITY f SUBTYPE OF (d); END_ENTITY;
ENTITY g SUBTYPE OF (d);
INVERSE
    attr3 : SET[2:4] OF d FOR attr2;
END_ENTITY;
ENTITY h SUBTYPE OF (d); END_ENTITY;
RULE f_attr1_big FOR (f);
WHERE
    WR1: SIZEOF (QUERY (fab <* f | fab.attr1 < 1000.0)) = 0;
END_RULE;
END_SCHEMA;
```

Тогда в словаре могут появиться следующие экземпляры **entity_definition**, закодированные в соответствии с ГОСТ Р ИСО 10303-21:

```
#1 = SCHEMA_DEFINITION('example_schema', $);
#100 = ENTITY_DEFINITION('a', (#101), #1, (), .F., .F., .T.);
#101 = WHERE_RULE('wr1', #100);
#102 = EXPLICIT_ATTRIBUTE('attr1', #100, #103, $, .F.);
#103 = REAL_TYPE($);
#110 = ENTITY_DEFINITION ('b', (), #1, (#100), .F., .T., .T.);
#120 = ENTITY_DEFINITION ('c', (), #1, (#100), .F., .T., .F.);
#121 = EXPLICIT_ATTRIBUTE('attr4', #120, #122, $, .F.);
#122 = STRING_TYPE($, .F.);
#123 = UNIQUENESS_RULE('ur1', (#121), #120);
#130 = ENTITY_DEFINITION ('d', (), #1, (#110), .F., .T., .T.);
#131 = EXPLICIT_ATTRIBUTE('attr2', #130, #160, $, .F.);
#140 = ENTITY_DEFINITION('e', (), #1, (#130), .F., .T., .T.);
#141 = EXPLICIT_ATTRIBUTE('attr2', #140, #142, #102, .F.);
#142 = INTEGER_TYPE();
#150 = ENTITY_DEFINITION('f', (), #1, (#130), .F., .T., .T.);
#160 = ENTITY_DEFINITION('g', (), #1, (#130), .F., .T., .T.);
#161 = INVERSE_ATTRIBUTE('attr3', #160, #130, $, #131, #162, #163, .F.);
#162 = INTEGER_BOUND(2);
#163 = INTEGER_BOUND(4);
#170 = ENTITY_DEFINITION('h', (), #1, (#130), .F., .T., .T.);
#200 = GLOBAL_RULE('f_attr1_big', (#150), (#201), #1);
#201 = WHERE_RULE('wr1', #200);
#300 = ENTITY_DEFINITION('b+c', (), #1, (#110, #120), .T., .T., .T.);
#301 = ENTITY_DEFINITION('e+g', (), #1, (#140, #160), .T., .T., .T.);
#302 = ENTITY_DEFINITION('e+h', (), #1, (#140, #170), .T., .T., .T.);
```

```
#303 = ENTITY_DEFINITION('e+g+h', (), #1, (#140, #160, #170), .T., .T., .T.);
#304 = ENTITY_DEFINITION('f+g', (), #1, (#150, #160), .T., .T., .T.);
#305 = ENTITY_DEFINITION('f+h', (), #1, (#150, #170), .T., .T., .T.);
#306 = ENTITY_DEFINITION('f+g+h', (), #1, (#150, #160, #170), .T., .T., .T.);
#307 = ENTITY_DEFINITION('c+d', (), #1, (#120, #130), .T., .T., .T.);
#308 = ENTITY_DEFINITION('c+e+g', (), #1, (#120, #140, #160), .T., .T., .T.);
#309 = ENTITY_DEFINITION('c+e+h', (), #1, (#120, #140, #170), .T., .T., .T.);
#310 = ENTITY_DEFINITION('c+e+g+h', (), #1, (#120, #140, #160, #170), .T., .T., .T.);
#311 = ENTITY_DEFINITION('c+f+g', (), #1, (#120, #150, #160), .T., .T., .T.);
#312 = ENTITY_DEFINITION('c+f+h', (), #1, (#120, #150, #170), .T., .T., .T.);
#313 = ENTITY_DEFINITION('c+f+g+h', (), #1, (#120, #150, #160, #170), .T., .T., .T.);
```

А.1.4 Границы и граничные выражения

Точность действительного (вещественного) типа; ширина строкового и двоичного типов, границы для типов набора (SET), списка (LIST), мультимножества (BAG); основные ограничения для инверсного атрибута и индексы для типа массива могут быть заданы в схеме целочисленными выражениями. Значения этих выражений могут быть основаны исключительно на схеме, внутри которой они объявлены, или могут зависеть от совокупности данной схемы. В случае, когда значение выражения основано исключительно на схеме, внутри которой оно объявлено, вычисленное целочисленное значение должно быть установлено в качестве значения атрибута **integer_bound.bound_value,** описывающего границу в словаре данных. В случае, когда значение выражения зависит от совокупности схемы, экземпляр **population_dependent_bound** должен объявлять границу в словаре данных.

А.1.5 Переобъявление атрибута

Когда атрибут в супертипе языка EXPRESS переопределяется в подтипе (см. 9.2.3.4 ГОСТ Р ИСО 10303-11), в словаре данных СИДД должны появляться отдельные экземпляры соответствующего подтипа атрибута, связывающие атрибут в супертипе и переопределенный атрибут в подтипе. Переопределенный атрибут должен быть элементом **entity_definition.attributes** для подтипа. Значение **attribute.redeclaring** в экземпляре атрибута для подтипа должно быть установлено ссылкой на экземпляр атрибута переобъявляемого супертипа.

А.2 Информация об эквивалентности области значений

А.2.1 Конструкции словаря

Эквивалентность областей значений двух типов данных объекта обеспечивается соответствующими определениями в схеме словаря СИДД. Тип объекта, существующий в одной схеме, может быть объявлен эквивалентным по области значений типу объекта из другой схемы при помощи типов данных объектов **domain_equivalent_type** и **external_schema**.

Эквивалентность областей значений задается попарно в тех определениях, связанных конкретным **schema_definition**, которые объявляют, что экземпляр объекта, основанный на собственной схеме, может ссылаться на экземпляр объекта, основанный на внешней схеме. СИДД-модели, основанные на внешней схеме, должны быть связаны с экземпляром схемы (**schema_instance**), основанным на собственной схеме в порядке получения этих ссылок. Однако экземпляры объекта, основанные на внешней схеме, не могут ссылаться на экземпляры объекта, основанные на собственной схеме, если нет дополнительного экземпляра **external_schema**, связанного с внешней схемой и определяющего **domain_equivalent_type** между внешней и собственной схемами.

Типы данных объекта **domain_equivalent_type** и **external_schema**, связанные с **schema_definition**, заполняются внешней системой, используя информацию, не определенную в настоящем стандарте. Данная информация (спецификации) может быть получена от разработчиков схем или определена объявлениями внутри самих схем.

А.2.2 Алгоритмы и методы объявления эквивалентности области значений

В настоящем пункте описаны два метода объявления эквивалентности по области значений в словаре данных СИДД:

1) с использованием EXPRESS-спецификации интерфейса;

2) с использованием заполнения файла по ГОСТ Р ИСО 10303-21 для части схемы словаря СИДД. Возможны и другие методы, но они не определены в настоящем стандарте.

Первый метод объявления эквивалентности области значений в схеме словаря СИДД обеспечивает алгоритм, основанный на EXPRESS-спецификации интерфейса. Два типа объявляются эквивалентными по области значений, если они выведены из одного и того же типа объекта с использованием конструкции USE или REFERENCE и одинаково использованы в двух схемах.

Пример 18

```
SCHEMA resource;
ENTITY identical;
        range : REAL;
END_ENTITY
ENTITY different;
```

```
        range : REAL;
END_ENTITY;
ENTITY between;
        range : REAL;
END_ENTITY;
END_SCHEMA;
SCHEMA ONE;
        USE FROM resource (identical, different, between);
RULE different_negative FOR (different);
WHERE
        WR1: SIZEOF (QUERY (a <* different | a.range < 0)) = 0;
END_RULE;
RULE between_small FOR (between);
WHERE
        WR1: SIZEOF (QUERY (a <* between | {0 < a.range < 10} )) = 0;
END_RULE;
END_SCHEMA;
SCHEMA TWO;
USE FROM resource (identical, different, between);
RULE different_positive FOR (different);
WHERE
        WR1: SIZEOF (QUERY (a <* different | a.range > 0)) = 0;
END_RULE;
RULE between_large FOR (between);
WHERE
        WR1: SIZEOF (QUERY (a <* between | {0 < a.range < 20} )) = 0;
END_RULE;
END_SCHEMA;
```

Из этих схем разработчик или пользователь СИДД может понять следующее:

- объект «identical» эквивалентен по области значений между схемами «ONE» и «TWO»;
- объект «different» не эквивалентен по области значений между схемами «ONE» и «TWO», так как его ограничения имеют разные области значений;
- объект «between» эквивалентен по области значений между схемами «TWO» и «ONE», но не наоборот, так как это ограничено возможностями экземпляров «between», принадлежащими схеме «ONE», быть допустимыми экземплярами «between» для схемы «TWO», но не наоборот.

Второй метод для объявления эквивалентности области значений заключается в обеспечении реализации информацией, закодированной согласно ГОСТ Р ИСО 10303-21 с использованием схемы словаря СИДД (см. раздел 6).

Пример 19 — Если разработчики или пользователи схем «ONE» и «TWO» хотят сделать «identical», «different» и «between» эквивалентными по области значений в СИДД данным способом, они могут добиться этого путем внесения следующей части данных в словарь:

```
/*schema_definition для схемы «ONE»
#1111 содержит определение объекта «identical» в схеме «ONE» и здесь не показано*/
#100 = SCHEMA_DEFINITION ('ONE');
#200 = EXTERNAL_SCHEMA ('TWO', (#300), #100);
#300 = DOMAIN_EQUIVALENT_TYPE ('identical', #1111);
/*schema_definition для схемы «TWO»
#6666 содержит определение объекта «identical» в схеме «TWO» и здесь не показано
#7777 содержит определение объекта «between» в схеме «TWO» и здесь не показано*/
#500 = SCHEMA_DEFINITION ('TWO');
#600 = EXTERNAL_SCHEMA ('ONE', (#700, #800), #500);
#700 = DOMAIN_EQUIVALENT_TYPE ('identical', #6666);
#800 = DOMAIN_EQUIVALENT_TYPE ('between', #7777);
```

ПРИЛОЖЕНИЕ В
(обязательное)

Форма заявки о соответствии реализации протоколу

Форма заявки о соответствии реализации протоколу (ЗСРП) обеспечивает оценку соответствия реализаций языковым привязкам СИДД, установленным в настоящем стандарте. Настоящее приложение имеет форму вопросника, который предназначен для заполнения разработчиком и может быть использован испытательной лабораторией при подготовке аттестационного тестирования.

Все реализаторы должны дать ответы на вопросы, приведенные в В.1 и В.2.

В.1 Соответствие заданной функции

В.1.1 Уровни транзакции

Какому уровню соответствует обеспечение? _____

указать один из следующих уровней

1 — нет транзакции;
2 — транзакции СИДД-модели;
3 — транзакции.

В.1.2 Уровни вычисления выражения

Какому уровню соответствует обеспечение? _____

указать один из следующих уровней

1 — нет вычисления;
2 — простое вычисление;
3 — сложное вычисление;
4 — полное вычисление.

В.1.3 Уровни записи событий сеанса

Какому уровню соответствует обеспечение? _____

указать один из следующих уровней

1 — нет записи;
2 — обеспечение записи.

В.1.4 Уровни области действия

Какому уровню соответствует обеспечение? _____

указать один из следующих уровней

1 — нет области действия;
2 — обеспечение области действия.

В.1.5 Уровни эквивалентности области значений

Какому уровню соответствует обеспечение? _____

указать один из следующих уровней

1 — нет эквивалентности области значений;
2 — эквивалентность области значений.

В.1.6 Обеспечиваемый класс реализации

Какому классу реализации соответствует обеспечение? _____

указать классы 1—7

В.2 Ограничения реализации

Что является основой для устанавливаемого времени [event.time]? _____

Какое максимальное число хранилищ может существовать внутри сеанса СИДД? _____

Какое максимальное число СИДД-моделей может существовать внутри хранилища? _____

Какая максимальная длина обеспечивается для типа STRING языка EXPRESS? _____

Какая максимальная длина обеспечивается для типа BINARY языка EXPRESS? _____

Какое ограничение точности обеспечивается для типа REAL языка EXPRESS? _____

Какое максимальное количество элементов может находиться в экземпляре агрегата переменной размерности? _____

Какое максимальное число индексных позиций может быть в экземпляре массива? _____

Если обеспечивается эквивалентность областей значений, какой метод используется для объявления ее в словаре данных? _____

ПРИЛОЖЕНИЕ С
(обязательное)

Регистрация информационного объекта

С.1 Обозначение документа

Для того чтобы обеспечить однозначное обозначение информационного объекта в открытой системе, настоящему стандарту присвоен идентификатор объекта

{ iso standard 10303 part(22) version(0) }

Смысл этого значения определен в ГОСТ Р ИСО/МЭК 8824-1 и описан в ГОСТ Р ИСО 10303-1.

С.2 Обозначение схемы

Для того чтобы обеспечить однозначное обозначение **SDAI_dictionary_schema** в открытой системе, **SDAI_dictionary_schema** (см. раздел 6) присвоен идентификатор объекта

{ iso standard 10303 part(22) version(0) object(1) SDAI-dictionary-schema(1) }

Смысл этого значения определен в ГОСТ Р ИСО/МЭК 8824-1 и описан в ГОСТ Р ИСО 10303-1.

Для того чтобы обеспечить однозначное обозначение **SDAI_session_schema** в открытой системе, **SDAI_session_schema** (см. раздел 7) присвоен идентификатор объекта

{ iso standard 10303 part(22) version(0) object(2) SDAI-session-schema(1) }

Смысл этого значения определен в ГОСТ Р ИСО/МЭК 8824-1 и описан в ГОСТ Р ИСО 10303-1.

Для того чтобы обеспечить однозначное обозначение **SDAI_population_schema** в открытой системе, **SDAI_population_schema** (см. раздел 8) присвоен идентификатор объекта

{ iso standard 10303 part(22) version(0) object(3) SDAI-population-schema(1) }

Смысл этого значения определен в ГОСТ Р ИСО/МЭК 8824-1 и описан в ГОСТ Р ИСО 10303-1.

Для того чтобы обеспечить однозначное обозначение **SDAI_parameter_data_schema** в открытой системе, **SDAI_parameter_data_schema** (см. раздел 9) присвоен идентификатор объекта

{ iso standard 10303 part(22) version(0) object(4) SDAI-parameter-data-schema(1) }

Смысл этого значения определен в ГОСТ Р ИСО/МЭК 8824-1 и описан в ГОСТ Р ИСО 10303-1.

ПРИЛОЖЕНИЕ D
(справочное)

Диаграммы на языке EXPRESS-G

Рисунки D.1 — D.10 соответствуют исходным текстам на языке EXPRESS, приведенным в разделах 6—9. В данных рисунках использована графическая нотация EXPRESS-G для языка EXPRESS. Нотация языка EXPRESS-G определена в приложении D ГОСТ Р ИСО 10303-11.

Рисунок D.1 — EXPRESS-G диаграмма 1 из 5-и схем словаря СИДД (см. также рисунки D.1—D.5)

Рисунок D.2 — EXPRESS-G диаграмма 2 из 5-и схем словаря СИДД (см. также рисунки D.1 и D.3—D.5)

Рисунок D.3 — EXPRESS-G диаграмма 3 из 5-и схем словаря СИДД (см. также рисунки D.1—D.5)

Рисунок D.4 — EXPRESS-G диаграмма 4 из 5-и схем словаря СИДД (см. также рисунки D.1—D.3 и D.5)

Рисунок D.5 — EXPRESS-G диаграмма 5 из 5-и схем словаря СИДД (см. также рисунки D.1—D.4)

Рисунок D.6 — EXPRESS-G диаграмма 1 из 2-х схем сеанса СИДД (см. также рисунок D.7)

Рисунок D.7 — EXPRESS-G диаграмма 2 из 2-х схем сеанса СИДД (см. также рисунок D.6)

Рисунок D.8 — EXPRESS-G диаграмма схемы совокупности СИДД

Рисунок D.9 — EXPRESS-G диаграмма 1 из 2-х схем параметризированных данных СИДД (см. также рисунок D.10)

Рисунок D.10 — EXPRESS-G диаграмма 2 из 2-х схем параметризированных данных СИДД
(см. также рисунок D.9)

ПРИЛОЖЕНИЕ Е
(справочное)

Распечатки (листинги) схем СИДД на языке EXPRESS

В настоящем приложении представлены листинги конструкций на языке EXPRESS, определенных в разделах 6—9 настоящего стандарта. Эти листинги не включают в себя текст или аннотации. Данное приложение предоставляется только в машинно-интерпретируемой форме по следующему адресу унифицированного указателя ресурсов (URL): http://www.nist.gov/sc4/step/parts/part022/current/part22.exp

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

диаграммы на языке EXPRESS-G . . . . . приложение D

Классы реализации

Реализации СИДД 13.1
уровни записи событий сеанса 13.1.3
уровни области действия 13.1.4
уровни эквивалентности области значений 13.1.5
уровни вычисления выражения для проверки и вычисляемых атрибутов 13.1.2
уровни транзакции 13.1.1
Спецификация классов реализаций 13.2
реализация класса 1 . . . . . 13.2.1
реализация класса 2 . . . . . 13.2.2
реализация класса 3 . . . . . 13.2.3
реализация класса 4 . . . . . 13.2.4
реализация класса 5 . . . . . 13.2.5
реализация класса 6 . . . . . 13.2.6
реализация класса 7 . . . . . 13.2.7
Команды, необходимые для классов реализаций 13.3

Команды СИДД

Команды агрегата прикладных экземпляров 10.13
задание значения текущего элемента 10.13.2
создание экземпляра агрегата в качестве текущего элемента 10.13.1
удаление текущего элемента 10.13.3
Команды агрегата экземпляров объекта 10.12
переход к новому текущему элементу 10.12.6
получение верхней границы 10.12.10
получение значения границы по итератору 10.12.8
получение количества элементов 10.12.1
получение нижней границы 10.12.9
получение текущего элемента 10.12.7
проверка на вхождение в экземпляр агрегата 10.12.2
создание итератора 10.12.3
удаление итератора 10.12.4
установка в начальное положение 10.12.5
Команды массива прикладных экземпляров 10.18
возврат к неустановленному значению по индексу 10.18.1
возврат текущего элемента к неустановленному значению 10.18.2
переиндексирование массива 10.18.3
переустановка индексов массива 10.18.4
Команды массива экземпляров объекта 10.17
получение верхнего индекса 10.17.4
получение нижнего индекса 10.17.3
проверка наличия значения по индексу 10.17.1
проверка текущего элемента 10.17.2
Команды неупорядоченного набора прикладных экземпляров 10.14
неупорядоченное добавление 10.14.1
неупорядоченное создание экземпляра агрегата 10.14.2
неупорядоченное удаление 10.14.3
Команды области действия 10.8
добавление к экспортному списку 10.8.5
копирование области действия 10.8.8
определение владельца области действия 10.8.2
получение области действия 10.8.3
пополнение области действия 10.8.1
проверка ссылочных ограничений области действия 10.8.9
удаление из области действия 10.8.4
удаление из экспортного списка 10.8.6
удаление области действия 10.8.7
Команды прикладного экземпляра 10.11
возврат атрибута в неустановленное значение 10.11.4

копирование прикладного экземпляра . . . 10.11.1
получение идентификатора сеанса . . . 10.11.7
получение описания . . . 10.11.8
получение постоянной метки . . . 10.11.6
проверка инверсных атрибутов . . . 10.11.11
проверка массива на наличие пустых элементов . . . 10.11.15
проверка наличия значений у явных атрибутов . . . 10.11.10
проверка правила «were» . . . 10.11.9
проверка размерности агрегата . . . 10.11.13
проверка ссылок явных атрибутов . . . 10.11.12
проверка точности действительного значения . . . 10.11.18
проверка уникальности агрегатов . . . 10.11.14
проверка ширины двоичного значения . . . 10.11.17
проверка ширины строки . . . 10.11.16
создание экземпляра агрегата . . . 10.11.5
удаление прикладного экземпляра . . . 10.11.2
установка значения атрибута . . . 10.11.3
Команды сеанса . . . 10.4
аварийное прерывание . . . 10.4.9
завершение доступа и фиксации транзакции . . . 10.4.10
завершение доступа к транзакции и аварийное прерывание . . . 10.4.11
закрытие сеанса . . . 10.4.4
запись ошибки . . . 10.4.1
запрос СИДД . . . 10.4.14
начало описания события . . . 10.4.2
начало транзакции с доступом «только чтение» . . . 10.4.7
начало транзакции с доступом «чтение—запись» . . . 10.4.6
окончание описания события . . . 10.4.3
открытие хранилища . . . 10.4.5
создание нефиксированного списка . . . 10.4.12
удаление нефиксированного списка . . . 10.4.13
фиксация транзакции . . . 10.4.8
Команды СИДД-модели . . . 10.7
завершение доступа «только чтение» . . . 10.7.5
завершение доступа «чтение—запись» . . . 10.7.7
начало доступа «только чтение» . . . 10.7.3
начало доступа «чтение—запись» . . . 10.7.6
отмена изменений . . . 10.7.10
перевод СИДД-модели в режим «чтение—запись» . . . 10.7.4
переименование СИДД-модели . . . 10.7.2
получение определения объекта . . . 10.7.8
создание экземпляра объекта . . . 10.7.9
сохранение изменений . . . 10.7.11
удаление СИДД-модели . . . 10.7.1
Команды списка прикладных экземпляров . . . 10.19
вставка перед текущим элементом . . . 10.19.1
вставка по индексу . . . 10.19.3
вставка после текущего элемента . . . 10.19.2
вставка экземпляра агрегата по индексу . . . 10.19.6
создание экземпляра агрегата перед текущим элементом . . . 10.19.4
создание экземпляра агрегата после текущего элемента . . . 10.19.5
удаление по индексу . . . 10.19.7
Команды среды . . . 10.3
открытие сеанса . . . 10.3.1
Команды типа . . . 10.9
получение определения сложного объекта . . . 10.9.1
проверка принадлежности к подтипу . . . 10.9.2
проверка принадлежности к подтипу СИДД . . . 10.9.3
проверка эквивалентности областей значений . . . 10.9.4
Команды упорядоченного набора прикладных экземпляров . . . 10.16
внесение значения по индексу . . . 10.16.1
создание экземпляра агрегата по индексу . . . 10.16.2

Команды упорядоченного набора экземпляров объектов . . . 10.15
переход в конец . . . 10.15.2
переход к предыдущему элементу . . . 10.15.3
получение значения границы по индексу . . . 10.15.4
получение по индексу . . . 10.15.1
Команды хранилища . . . 10.5
закрытие хранилища . . . 10.5.3
создание СИДД-модели . . . 10.5.1
создание экземпляра схемы . . . 10.5.2
Команды экземпляров объектов . . . 10.10
определение соответствия экземпляра заданному типу . . . 10.10.5
определение соответствия экземпляра типу прикладной схемы . . . 10.10.6
определение соответствия экземпляра типу прикладной схемы и схемы параметризированных данных СИДД . . . 10.10.7
поиск пользователей экземпляра объекта . . . 10.10.8
поиск пользователей экземпляра объекта в заданной роли . . . 10.10.9
поиск ролей экземпляра . . . 10.10.11
поиск СИДД-модели экземпляра объекта . . . 10.10.3
поиск типов данных экземпляра . . . 10.10.12
получение значения атрибута . . . 10.10.1
получение значения границы атрибута . . . 10.10.10
получение типа экземпляра . . . 10.10.4
проверка атрибута . . . 10.10.2
Команды экземпляра схемы . . . 10.6
добавление СИДД-модели . . . 10.6.3
определение актуальности проверки . . . 10.6.9
переименование экземпляра схемы . . . 10.6.2
проверка глобального правила . . . 10.6.5
проверка области значений ссылки на экземпляр . . . 10.6.7
проверка правила уникальности . . . 10.6.6
проверка экземпляра схемы . . . 10.6.8
удаление СИДД-модели . . . 10.6.4
удаление экземпляра схемы . . . 10.6.1

Модели состояний СИДД

Модель состояния для транзакции уровня 1 . . . 12.1
переходы состояний . . . 12.1.6
состояние «Начало доступа к СИДД-модели в режиме «только чтение» 1» . . . 12.1.4
состояние «Начало доступа к СИДД-модели в режиме «чтение—запись» 1» . . . 12.1.5
состояние «Нет сеанса 1» . . . 12.1.1
состояние «Открытое хранилище 1» . . . 12.1.3
состояние «Сеанс 1» . . . 12.1.2
Модель состояния для транзакции уровня 2 . . . 12.2
переходы состояний . . . 12.2.6
состояние «Начало доступа к СИДД-модели в режиме «только чтение» 2» . . . 12.2.4
состояние «Начало доступа к СИДД-модели в режиме «чтение—запись» 2» . . . 12.2.5
состояние «Нет сеанса 2» . . . 12.2.1
состояние «Открытое хранилище 2» . . . 12.2.3
состояние «Сеанс 2» . . . 12.2.2
Модель состояния для транзакции уровня 3 . . . 12.3
переходы состояний . . . 12.3.11
состояние «Начало доступа к модели в режиме «только чтение» в хранилище с доступом «только чтение» 3» . . . 12.3.8
состояние «Начало доступа к модели в режиме только чтение» в хранилище с доступом «чтение—запись» 3» . . . 12.3.9
состояние «Начало доступа к модели в режиме «чтение—запись» в хранилище с доступом «чтение—запись» 3» . . . 12.3.10
состояние «Начало транзакции в режиме «чтение—запись» 3» . . . 12.3.4
состояние «Начало транзакции в режиме «только чтение» 3» . . . 12.3.3
состояние «Нет сеанса 3» . . . 12.3.1
состояние Открытое хранилище 3» . . . 12.3.5
состояние «Открытое хранилище в режиме «только чтение» 3» . . . 12.3.6
состояние Открытое хранилище в режиме «чтение—запись» 3» . . . 12.3.7
состояние «Сеанс 3» . . . 12.3.2

Отображение EXPRESS-конструкции в схемы словаря СИДД — приложение А
конструкции языка EXPRESS — А 1
информация об эквивалентности области значении — А 2
Ошибки СИДД — 11
Схемы СИДД
Схема параметризированных данных СИДД — 9
Объекты схемы параметризированных данных СИДД — 9 4
aggregate_instance — 9 4 11
application_indexed_array_instance — 9 4 20
application_instance — 9 4 3
array_instance — 9 4 19
attribute_value — 9 4 7
bag_instance — 9 4 14
dictionary_instance — 9 4 5
entity_instance — 9 4 2
enumeration_value — 9 4 10
iterator — 9 4 1
list_instance — 9 4 16
non_persistent_list_instance — 9 4 18
ordered_collection — 9 4 15
schema_defined_list_instance — 9 4 17
sdai__instance — 9 4 4
select_aggregate_instance — 9 4 9
select_value — 9 4 8
session_instance — 9 4 6
set_instance — 9 4 13
unordered_collection — 9 4 12
Типы схемы параметризированных данных СИДД — 9 3
aggregate_primitive — 9 3 3
assignable_primitive — 9 3 2
binary_value — 9 3 5
boolean_value — 9 3 9
bound_instance_value — 9 3 11
integer_value — 9 3 6
logical_value — 9 3 10
number_value — 9 3 8
primitive — 9 3 1
query_source — 9 3 12
real_value — 9 3 7
string_value — 9 3 4
Схема сеанса СИДД — 7
Объекты схемы сеанса СИДД — 7 4
error_event — 7 4 7
event — 7 4 6
implementation — 7 4 2
sdai_repository — 7 4 3
sdai_repository_contents — 7 4 4
sdai_session — 7 4 1
sdai_transaction — 7 4 5
Типы схемы сеанса СИДД — 7 3
access_type — 7 3 1
error_base — 7 3 2
time_stamp — 7 3 3
Схема словаря СИДД — 6
Объекты схемы словаря СИДД — 6 4
aggregation_type — 6 4 30
array_type — 6 4 35
attribute — 6 4 13
bag_type — 6 4 33
binary_type — 6 4 25
boolean_type — 6 4 27

bound 6 4 36
defined_type 6 4 11
derived_attribute 6 4 14
domain_equivalent_type 6 4 9
entity_definition 6 4 12
enumeration_type 6 4 28
explicit_attribute 6 4 15
explicit_item_id 6 4 4
external_schema 6 4 8
global_rule 6 4 19
implicit_item_id 6 4 7
integer_bound 6 4 38
integer_type 6 4 22
interface_specification 6 4 2
interfaced_item 6 4 3
inverse_attribute 6 4 16
list_type 6 4 34
logical_type 6 4 26
named_type 6 4 10
number_type 6 4 21
population_dependent_bound 6 4 37
real_type 6 4 23
referenced_item 6 4 6
schema_definition 6 4 1
select_type 6 4 29
set_type 6 4 32
simple_type 6 4 20
string_type 6 4 24
uniqueness_rule 6 4 17
used_item 6 4 5
variable_size_aggregation_type 6 4 31
where_rule 6 4 18
Типы схемы словаря СИДД 6 3
base_type 6 3 1
constructed_type 6 3 2
explicit_or_derived 6 3 5
express_id 6 3 6
info_object_id 6 3 7
type_or_rule 6 3 4
underlying_type 6 3 3
Схема совокупности СИДД 8
Объекты схемы совокупности СИДД 8 4
entity_extent 8 4 4
schema_instance 8 4 1
scope 8 4 5
sdai_model 8 4 2
sdai_model_contents 8 4 3
Типы схемы совокупности СИДД 8 3
entity_definition 8 3 2
schema_definition 8 3 1
Форма заявки о соответствии реализации протоколу (ЗСРП) приложение В

УДК 656.072:681.3:006.354 ОКС 25.040.40 П87 ОКСТУ 4002

Ключевые слова: автоматизация, средства автоматизации, прикладные автоматизированные системы, промышленные изделия, данные, представление данных, обмен данными, машинные интерфейсы, реализация

Редактор *В. Н. Огурцов*
Технический редактор *Л. А. Гусева*
Корректор *Н. И. Гаврищук*
Компьютерная верстка *Т. Ф. Кузнецовой*

Изд. лиц. № 02354 от 14.07.2000. Сдано в набор 06.08.2002. Подписано в печать 28.10.2002. Усл. печ. л. 15,81. Уч.-изд. л. 15,75.
Тираж 351 экз. С 7933. Зак. 1907

ИПК Издательство стандартов,107076 Москва, Колодезный пер., 14.
http.//www.standards.ru e-mail: info@standards.ru
Набрано в Калужской типографии стандартов на ПЭВМ.
Калужская типография стандартов, 248021 Калуга, ул. Московская, 256.
ПЛР № 040138