УДК 004.052:004.738 DOI: 10.31799/2077-5687-2026-3-17-23

# ДИАГНОСТИКА И ЛОКАЛИЗАЦИЯ ОШИБОК ИДЕНТИФИКАЦИОННЫХ ДАННЫХ ПРИ МЕЖСИСТЕМНОМ ОБМЕНЕ НА ОСНОВЕ CCSDS

**В. Е. Таратун**
Санкт-Петербургский государственный университет аэрокосмического приборостроения

*Предложена процедура разбора RFID-идентификаторов CCSDS перед их использованием в системе учёта. Она связывает результат отдельной проверки с исходной записью, байтовой позицией, логическим полем и источником ожидаемого значения. Нарушение формата, расхождение с контрольной записью и невозможность выполнить проверку получают разные статусы. Для сопоставления записей заданы требования к внешнему ключу и пригодности эталона. Процедура реализована в демонстрационной программе на Python; воспроизведены восемь заданных примеров и выполнены проверки полного множества однобайтовых кодов во всех символьных позициях. Установлены границы обнаружения допустимых подмен: без контрольных сведений они проходят проверку кодирования. Результат предназначен для воспроизводимого разбора несоответствий. Соответствие всей системы стандарту и сокращение трудозатрат оператора в этой проверке не оценивались.*

*Ключевые слова: радиочастотная идентификация, CCSDS, межсистемный обмен, идентификационные данные, декодирование, диагностика ошибок, локализация ошибок.*

***Для цитирования:***
*Таратун, В. Е. Диагностика и локализация ошибок идентификационных данных при межсистемном обмене на основе CCSDS / В. Е. Таратун // Системный анализ и логистика. – 2026. – № 3(51). – с. 17-23. DOI: 10.31799/2077-5687-2026-3-17-23.*

# DIAGNOSIS AND LOCALIZATION OF IDENTIFICATION DATA ERRORS IN CCSDS-BASED INTERSYSTEM EXCHANGE

**V. E. Taratun**
St. Petersburg State University of Aerospace Instrumentation

*A procedure is proposed for inspecting CCSDS RFID identifiers before their use in an inventory system. Each check result is linked to the original record, byte position, logical field and source of the expected value. Format violations, differences from a reference record and checks that cannot be performed receive distinct statuses. Record comparison requires an external matching key and a usable reference. A demonstration implementation in Python reproduces eight constructed cases and checks every byte value at each character position. The results delimit the detection of valid-code substitutions: without reference information, such substitutions pass encoding checks. The procedure supports reproducible investigation of discrepancies. System-wide conformance and reductions in operator effort were not evaluated.*

*Keywords: radio frequency identification, CCSDS, intersystem exchange, identification data, decoding, error diagnosis, error localization.*

***For citation:***
*Taratun, V. E. Diagnosis and Localization of Identification Data Errors in CCSDS-Based Intersystem Exchange / V. E. Taratun // System analysis and logistics. – 2026. – № 3(51). – p. 17-23. DOI: 10.31799/2077-5687-2026-3-17-23.*

## Введение

При обмене между системами учёта отказ в обработке идентификатора может иметь разные основания. Код символа бывает недопустимым, допустимое значение — отличным от контрольного, а сведения о соответствующем объекте — недоступными получателю. Для разработчика это разные задачи. Исправление кодирования не устраняет отсутствие согласованной учётной записи; совпадение формата не подтверждает, что приняты данные нужного объекта.

В работе рассматривается запись после её получения программой и до подтверждения учётной операции. Цель — задать результат диагностики, по которому можно повторить выполненную проверку и установить предел её вывода. Процедура использует фиксированную редакцию CCSDS 881.1-B-1 [1]. Контроль радиоканала, привязка RFID-метки



к физическому объекту и поиск причины программного дефекта требуют дополнительных наблюдений и в эту задачу не входят.

**1. Основания процедуры**

Н. Н. Майоров и В. Е. Таратун [2] описали модель представления идентификатора в системе учёта и программные модули кодирования и декодирования на Java. В следующей работе [3] рассмотрены обмен файлами между независимыми реализациями и выдача сообщений об ошибочных полях. Доклад NASA [4, слайд 7] фиксирует распределение работ над двумя прототипами между NASA и FSA. Обнаружение несоответствий при таком обмене уже относится к предшествующим результатам [3]. В настоящей статье конкретизируются состав диагностической записи и условия, при которых результат отдельной проверки считается обоснованным.

Дифференциальное тестирование сопоставляет результаты независимых реализаций на связанных входах. А. Гроус, Г. Хольцман и Р. Джоши [5] применяли такое сравнение при разработке файловой системы и учитывали допустимые различия поведения. Для рассматриваемой задачи существенны два условия: записи должны относиться к одному испытательному случаю, а ожидаемое представление должно иметь независимое основание. Само несовпадение двух программ указывает на расхождение; чтобы признать ошибочной одну из них, требуется проверка относительно принятой спецификации.

Предлагаемая процедура объединяет проверку внешней записи, допустимости символьных кодов, справочных назначений и совпадения с эталоном. Для каждого уровня сохраняется собственный статус. Прикладной вклад состоит в однозначной привязке сообщения к полю и координатам, а также в обработке непригодных контрольных данных и зависимых пропусков. Байтовое расположение полей и набор допустимых кодов заимствованы из [1]; профиль входа и организация диагностических сообщений заданы в этой работе.

**2. Структура записи и основания проверки**

Идентификатор содержит 96 бит, то есть 12 байтов. В табл. 1 приведено соответствие между байтами и полями по [1, табл. 3-1, рис. 3-2]. Байтовые позиции отсчитываются от нуля; позиции внутри символьного поля — от единицы. Object-ID задаёт класс объектов, а не уникальный номер физического экземпляра.

Таблица 1 – Координаты полей в 12-байтовой записи

| **Поле** | **Байты** | **Правило локализации** |
|---|---|---|
| Database-ID | 0 | Один символ; позиция в поле 1 |
| Owner-ID | 1 и 9 | Байт 1 — символ 1; байт 9 — символ 2 |
| Program-ID | 2 и 3 | Позиция символа равна номеру байта минус 1 |
| Serial-ID | 4–8 | Позиция символа равна номеру байта минус 3 |
| Object-ID | 10 и 11 | Первый и второй байты числа; единица позиции — байт |

Символьные поля используют ECMA-113/ISO 8859-5; для байтов 0–9 допустимы коды 20h–7Eh и A0h–FFh [1, § A2.2(b), табл. A-1]. Каждое значение сравнивается с этими границами до отображения символа. Код Object-ID вычисляется как $256b_{10} + b_{11}$, где $b_{10}$ и $b_{11}$ — численные значения соответствующих байтов; порядок принят по рис. 3-2 [1]. После выделения двух байтов результат автоматически находится в пределах 0–65535. Эта граница не обнаружит перестановку байтов: 0001h и 0100h задают допустимые числа 1 и 256.

Вход демонстрационной процедуры — строка ровно из 24 знаков алфавита 0–9, A–F, a–f, без префикса и разделителей. Это локальный профиль интерфейса. Проверяется исходная строка: пробелы и переносы строк не удаляются, недостающие знаки не дополняются. Для

иных представлений нужен отдельный адаптер с описанным преобразованием и сохранением исходных координат; его реализация здесь не рассматривается. Если неверна длина либо встретился знак вне алфавита, модуль сообщает нарушение профиля и прекращает разбор на байты. Поле повреждения по строке с неизвестными границами не назначается.

Допустимость кода и назначение значения проверяются раздельно. Database-ID должен соответствовать назначению реестра SANA [1, § A2.2(c); 6]. Для Object-ID стандарт различает область 0–32767, используемую одной стороной, и область 32768–65535 для общего администрирования [1, § A2.2(e–f); 7]. Попадание в верхнюю область не означает, что значение уже назначено. Проверка учитывает статус записи реестра и её назначение. Выбор требуемой области определяется контекстом применения, который нельзя восстановить только по двум байтам.

При справочной проверке сохраняются снимок реестра, его дата или версия, проверяемое значение и критерий принятия. Допустима неизменяемая доступная ссылка на содержимое снимка; одной даты недостаточно для повторения поиска. Отсутствие снимка, отсутствие назначения в доступном снимке и отсутствие объекта в учётной базе дают разные сообщения. Если неизвестно, требуется ли общий либо локальный класс, проверка области применения остаётся невыполненной. В демонстрационной программе ниже реестры и учётная база не подключаются; её результаты относятся к профилю, кодам и сравнению записей.

**3. Диагностическое сообщение и последовательность действий**

Модуль возвращает исходную запись и результаты отдельных правил. Используются четыре статуса: «выполнено» — проверка пройдена; «нарушение» — вход противоречит проверенному ограничению; «расхождение» — байты отличаются от пригодного эталона; «не выполнено» — отсутствует необходимое основание либо не выполнена предшествующая проверка. Последний статус сопровождается причиной. Уровень пройден, только если пройдены все входящие в него проверки; обнаруженные нарушения и пропуски сохраняются раздельно. Структура сообщения приведена в табл. 2.

Таблица 2 – Сведения для воспроизведения диагностического вывода

| **Группа сведений** | **Содержание** |
|---|---|
| Контекст | Внешний ключ случая; версии схемы, профиля и проверяющего модуля |
| Вход | Исходная строка; байты, если преобразование выполнено |
| Координаты | Позиции знаков строки; номер байта; имя поля; позиция в поле и её единица |
| Проверка | Идентификатор правила; статус; причина нарушения или пропуска |
| Наблюдение | Принятое значение в числовой форме; декодированный символ при допустимом коде |
| Основание | Ограничение спецификации; полный эталон или неизменяемая доступная ссылка, внешний ключ и версия; ожидаемый диапазон или байт |

1. Сохранить вход вместе с внешним ключом. Проверить длину и алфавит. Нарушение длины фиксируется фактическим значением, нарушение алфавита — позицией каждого недопустимого знака. Зависимые проверки получают статус «не выполнено».
2. Преобразовать допустимую строку в 12 байтов и проверить все десять символьных позиций. После первого недопустимого кода обход продолжается, поскольку



границы байтов уже установлены. Каждое нарушение получает координаты по табл. 1. Недопустимый код сохраняется числом, без замены на знак вопроса.

3. При подключённых справочниках проверить назначение значений. Поиск, зависящий от недопустимого поля, пропустить с указанием причины. Например, отрицательный ответ поиска по ошибочному Database-ID не является самостоятельным подтверждением отсутствия базы данных.
4. Проверить пригодность контрольной записи и её привязку, затем сравнить байты. Если эталона нет, он нарушает профиль или ограничения символьных кодов либо его внешний ключ отличается, сравнение не выполняется. Эти причины относятся к контрольным данным и не выдаются за расхождение принятой записи.
5. Сохранить все результаты и передать их системе учёта. Исправление входа не выполняется диагностическим модулем. Если уполномоченный участник представит новую запись, она проверяется отдельно с сохранением связи с предыдущей версией.

Внешний ключ связывает записи независимо от проверяемого идентификатора: например, это номер тестового случая, выданный до обмена. Поиск эталона только по принятому идентификатору потеряет такую связь при его подмене. Проверка формата эталона также не делает его истинным: правильность исходных полей и доверие к контрольному источнику задаются организатором испытания. Равенство байтов означает совпадение двух представлений при этих условиях, но не доказательство соответствия физическому объекту.

Пригодный эталон позволяет сравнивать байты даже там, где принятая запись содержит недопустимый символьный код. В этом случае сохраняются два основания: нарушение диапазона и отличие от контрольного байта. Они не суммируются как две ошибочные записи. Для допустимого кода, заменённого другим допустимым кодом, остаётся только второе основание. Разный регистр букв шестнадцатеричной строки расхождения не создаёт, если полученные байты одинаковы.

Для байта с номером i соответствующая пара знаков занимает позиции 2i + 1 и 2i + 2 внешней строки. Например, байт 5 связан со знаками 11–12 и вторым символом Serial-ID. Второй символ Owner-ID находится в байте 9 и связан со знаками 19–20. У Object-ID сохраняются номера первого или второго байта числа; символьная позиция этому полю не приписывается.

**4. Воспроизводимая проверка на заданных данных**

Контрольная запись 4941435031323334354200 01 задаёт Database-ID = I, Owner-ID = AB, Program-ID = CP, Serial-ID = 12345 и Object-ID = 1. Это условные данные. Они не относятся к реальному грузу и не воспроизводят файл NASA. Восемь случаев К0–К7 образованы изменением строки либо доступности эталона; результаты демонстрационной реализации приведены в табл. 3. Эталон и принятый вход связываются одним внешним ключом, кроме специально заданных дополнительных проверок.

Таблица 3 – Результаты восьми контрольных случаев

| **Случай** | **Условие** | **Результат выполненных проверок** |
|---|---|---|
| К0 | Исходная строка; эталон задан | Профиль и коды допустимы; различий байтов нет |
| К1 | Удалён последний знак | Длина 23 вместо 24; байтовые проверки не выполнены |
| К2 | Знак 9 заменён на G | Нарушение алфавита в позиции 9; байтовые проверки не выполнены |
| К3 | Байт 5: 32h → 00h; эталон задан | Нарушение диапазона; Serial-ID, символ 2; расхождение с 32h в байте 5 |
| К4 | Байт 9: 42h → 7Fh; эталон задан | Нарушение диапазона; Owner-ID, символ 2; расхождение с 42h в байте 9 |
| К5 | Байт 5: 32h → 39h; эталона нет | Код допустим; сравнение не выполнено из-за отсутствия эталона |
| К6 | Вход К5; эталон задан | Код допустим; Serial-ID, символ 2: получено 9 вместо контрольного 2 |
| К7 | Object-ID: 0001h → 0100h | Число 256 вместо контрольного 1; расхождения в байтах 10 и 11 |

Сопоставление К5 и К6 отделяет свойства входа от доступных сведений о нём. Входные байты одинаковы, и оба раза символьная проверка проходит. Только добавление эталона даёт основание отметить расхождение. К3 иллюстрирует независимость проверок: удаление эталона уберёт ожидаемый байт 32h, но не отменит нарушения символьного диапазона кодом 00h. Ни одно из этих сообщений не разрешает автоматически подставить контрольный байт вместо принятого.

Программа выполнена в Python 3.12.14. Перебраны 256 кодов в каждой из десяти символьных позиций: 2560 вариантов, из них 650 с запрещённым кодом и 1910 с разрешённым. Все запрещённые коды обнаружены и локализованы, разрешённые не вызвали нарушения диапазона. Для Object-ID проверены все 65 536 значений: результаты декодирования совпали с вычислением по старшему и младшему байтам; числовые байты не дали ложных символьных нарушений. Кроме восьми основных случаев выполнены 17 дополнительных. Эти проверки охватывают однобайтовые значения и заданные граничные ситуации, но не все сочетания полей, сбои передачи и варианты интеграции.

Дополнительные случаи проверяют две особенности эталона: непригодный контрольный вход не создаёт ложного расхождения, а другой внешний ключ блокирует сравнение. Подмена латинской A кодом кириллической А в Owner-ID проходит проверку символьного диапазона и обнаруживается только при сравнении с исходным эталоном. Обе буквы могут выглядеть одинаково; диагностическое основание задают байты 41h и B0h.

### 5. Применение сообщения и границы локализации

Для учётной операции заранее фиксируется набор обязательных правил, включающий как минимум профиль и символьные ограничения. Подтверждение допускается только тогда, когда каждое обязательное правило имеет статус «выполнено». Отсутствие его результата, «нарушение», «расхождение» или «не выполнено» блокируют подтверждение. Причина остаётся доступной оператору: требуется исправить источник данных, согласовать контрольные сведения либо выполнить отсутствующую проверку. Необязательные результаты сохраняются, но не изменяют заданное правило допуска. Организацию такого допуска и её влияние на работу оператора демонстрационная программа не проверяет.

Для поиска места возникновения дефекта нужны точки наблюдения, связанные тем же

внешним ключом: исходные поля, байты после кодирования, байты перед декодированием и восстановленные поля. Побайтово сравниваются выход кодировщика и вход декодировщика. Переходы между полями и байтами проверяются по независимо заданному преобразованию спецификации. Так устанавливается первое наблюдаемое расхождение между точками регистрации. Разные выходные и входные байты требуют проверки передачи, преобразований файла и самого журналирования. Они ещё не доказывают, какая из этих операций внесла изменение.

Согласованная ошибка кодировщика и декодировщика может сохранять исходные поля после полного цикла. Независимое контрольное значение обнаружит её лишь на входе, который проявляет ошибочную интерпретацию, и при корректности самого эталона. Поэтому совпадение одного случая не распространяется на остальные значения и ветви алгоритма. Опубликованные ранее агрегированные результаты обмена [3] также не заменяют журналов отдельных записей: по ним нельзя восстановить байтовую причину каждого отказа.

**Заключение**

Получена процедура, в которой диагностический результат сохраняет координаты наблюдаемого нарушения и достаточное основание для его повторной проверки. Контрольные случаи показывают принципиальную границу: допустимая подмена не обнаруживается проверкой кодирования, а непригодный эталон не даёт основания признать принятую запись ошибочной. При известных границах 12-байтовой записи все недопустимые символьные коды локализуются по заданному соответствию полей. Этот результат относится к проверенным правилам демонстратора. Для вывода о причине исторических ошибок, соответствии всей системы CCSDS или сокращении времени диагностики нужны иные данные и испытания.

## СПИСОК ЛИТЕРАТУРЫ


1. Spacecraft Onboard Interface Services—RFID Tag Encoding Specification. CCSDS 881.1-B-1 [Электронный ресурс] // Recommended Standard. Issue 1. October 2017. – URL: https://ccsds.org/Pubs/881x1b1.pdf (дата обращения: 11.09.2026).
2. *Майоров Н. Н.* Исследование и практическая реализация модели идентификации объектов систем учета материального потока на основе стандартов CCSDS / Н. Н. Майоров, В. Е. Таратун // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. – 2022. – № 1. – С. 72–80. DOI: 10.24143/2072-9502-2022-1-72-80.
3. *Майоров Н. Н.* Исследование информационного взаимодействия между различными подсистемами на основе стандартов CCSDS при идентификации материальных объектов / Н. Н. Майоров, В. Е. Таратун // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. – 2022. – № 2. – С. 52–65. DOI: 10.24143/2072-9502-2022-2-52-65.
4. *Kearney M.* NASA Report to the CCSDS [Электронный ресурс] // Presentation. NASA Technical Reports Server. Document ID 20150002580. Management Council. London, November 2014. – URL: https://ntrs.nasa.gov/citations/20150002580 (дата обращения: 11.09.2026).
5. *Groce A., Holzmann G., Joshi R.* Randomized Differential Testing as a Prelude to Formal Verification // Proceedings of the 29th International Conference on Software Engineering (ICSE 2007). – Pasadena, 2007. – 10 p. DOI: 10.1109/ICSE.2007.68.
6. Space Assigned Numbers Authority [Электронный ресурс] // RFID Tag-Encoding Database-ID. 2017. – URL: https://sanaregistry.org/r/rfid_database_ids/ (дата обращения: 11.09.2026).
7. Space Assigned Numbers Authority. [Электронный ресурс] // RFID Tag-Encoding Object-ID. 2017. – URL: https://sanaregistry.org/r/rfid_object_ids/ (дата обращения: 11.09.2026).



## ИНФОРМАЦИЯ ОБ АВТОРЕ

**Таратун Виталий Евгеньевич**
Канд. техн. наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения
Россия, 190000, Санкт-Петербург, ул. Большая Морская, д.67, лит. А
E-mail: taratun.vitaliy@gmail.com

## INFORMATION ABOUT THE AUTHOR

**Taratun Vitaly Evgenievich**
Candidate of Technical Sciences, Associate Professor
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: taratun.vitaliy@gmail.com

*Дата поступления: 13.09.2026*
*Дата принятия: 13.09.2026*