# Документация

# JSON в дерево

Компонент преобразует данные из формата JSON в иерархическую древовидную структуру, обеспечивая контроль соответствия входных данных ожидаемой структуре (проверка типов данных, обязательных узлов и недопустимых полей), поддержку шаблонов для распознавания строк как Дата/время, а также гибкую настройку выходного дерева в автоматическом или ручном режиме.

## Порты

### Вход

**Входной набор данных** — таблица, в которой есть столбец со значениями, записанными в формате JSON.

### Выход

**Выходное дерево** — иерархическая структура данных дерева, соответствующая разобранному JSON.

## Мастер настройки

### Основные параметры

**Имя столбца** — имя столбца, содержащего JSON, в форме выпадающего списка доступных столбцов строкового типа (обязательное поле).

**Формат даты и времени** — шаблон для распознавания дат в JSON. Поскольку JSON не имеет специального типа для дат (значения передаются как строки), этот параметр позволяет указать, как именно следует интерпретировать строковые данные при преобразовании в Дата/время. По умолчанию используется стандарт ISO 8601. Доступны предустановленные шаблоны и возможность ручного ввода.

**Исключить корневой узел** — если опция включена, корневой узел исключается из результата, но только в том случае, если у него ровно один дочерний узел.

> **Примечание:** Если у корневого узла несколько дочерних узлов или корневой узел является массивом, то он не будет исключен.

**Строгое соответствие структуры** — проверка структуры выполняется всегда, независимо от состояния опции. Разница заключается в реакции на критические несоответствия:

- Если включено (используется по умолчанию), несоответствие между JSON и структурой выходного дерева вызовет ошибку в случаях:
  - Если в JSON присутствует узел, отсутствующий в структуре дерева.
  - Если JSON содержит массив, но в дереве узел объявлен как примитив.
  - Если в JSON узел является примитивом, но в дереве объявлен как контейнер.
- Если выключено, ошибки записываются в лог.

При включённом режиме **Автоопределение структуры** выходное дерево строится на основе JSON, а при выключенном — структура дерева задаётся вручную или импортируется (аналогично компоненту Таблица в дерево).

**Корневой узел** — эта функция доступна при автоматическом определении структуры выходного дерева. Она позволяет выбрать способ определения массива у корневого узла:

- **Автоопределение (по умолчанию)** — корневой узел получает признак массива, если во входном наборе данных несколько строк.
- **Всегда массив** — корневой узел всегда будет массивом, даже если входной JSON всего один и не является массивом.
- **Всегда не массив** — производится проверка, если входной JSON не является массивом и содержит только одну запись, система выдаст ошибку.

# Структура выходного дерева

## Задание структуры дерева вручную

Настройка дерева доступна при отключённом параметре *Определять структуру автоматически*.

Ручное редактирование структуры выполняется с помощью инструментов на панели управления или в контекстном меню узла.

Панель инструментов включает команды:

- **Добавить дочерний узел** — позволяет добавить для текущего узла подчиненный.
- **Добавить соседний узел** — позволяет создать узел того же уровня иерархии, что и выбранный.
- **Редактировать...** — позволяет вызвать окно редактирования и изменить значения атрибутов для выбранного узла (команду также можно вызвать горячей клавишей *F2*).
- **Переместить вверх** и **Переместить вниз** — позволяют менять порядок узлов, при этом корневой узел **Root** переместить нельзя (команды также доступны по комбинации горячих клавиш *Ctrl+Up* и *Ctrl+Down* соответственно).
- **Загрузить из XSD...** — загружает структуру узлов выходного дерева из файла формата XSD.

- **Загрузить из JSON...** — загружает структуру дерева напрямую из JSON-документа.
- **Удалить** — удаляет дочерний узел дерева, иконка этой команды высвечивается при наведении курсора на узел (команду также можно вызвать горячей клавишей *Delete*).
- **Удалить все...** — удаляет все дочерние узлы (комбинация горячих клавиш *Shift+Delete*).

> **Примечание:** Удалить корневой узел **Root** нельзя.

При выполнении команд **Добавить дочерний узел**, **Добавить соседний узел**, **Редактировать...** задаются значения атрибутов:

- **Имя** — уникальное наименование узла в рамках одного набора данных. Может состоять из:
  - заглавных или строчных латинских букв;
  - символа подчеркивания;
  - цифр (не может быть первым символом).
- **Имя JSON поля** — исходное название свойства (ключа) в JSON-документе. Должно в точности соответствовать ключу в исходных данных для обеспечения корректного сопоставления.
- **Тип данных** — один из возможных типов данных.
- **Вид данных** — один из возможных видов данных.

Кроме того, можно установить следующие признаки:

- **Массив** — при установке флага выбранный узел будет определен как упорядоченное множество (массив) данных одного типа.
- **Контейнер** — если флаг установлен, то для выбранного узла можно создать дочерние.

Записи в списках *Входное дерево* и *Выходное дерево* можно отфильтровать с помощью команды **Фильтрация** на соответствующей панели инструментов.

## Загрузка структуры узлов из XSD схемы

Структуру выходного дерева можно загрузить из файла формата XSD с помощью команды **Загрузить из XSD...**

В появившемся диалоговом окне необходимо заполнить поля:

- **XSD файл** — поле для выбора файла (не редактируемое).
- **Пространство имен** — выбор пространства имен из списка всех пространств имен описанных в файле XSD, ограничивает выбор корневого элемента только указанным пространством. По умолчанию принимает значение *Все пространства имен*.
- **Корневой элемент** — выбор корневого узла из списка узлов загружаемой xml структуры. По умолчанию имеет значение корневого узла в этой структуре.
- **Глубина рекурсии** — максимальное число рекурсий при раскрытии рекурсивных узлов. Выбирается в диапазоне от 0 до 3. По умолчанию равно 1, что означает: каждый рекурсивный узел будет раскрыт автоматически, но рекурсивные узлы внутри этих узлов останутся нераскрытыми. Дополнительно раскрытие рекурсивных узлов можно проводить вручную после построения дерева. При значении 0 все рекурсивные узлы не будут раскрыты автоматически.

После заполнения всех полей необходимо нажать кнопку **Загрузить**, и XSD схема будет загружена для дальнейшей работы.

> **Важно:** При загрузке схемы существующие узлы будут удалены.

### Загрузка дерева из JSON

Структуру выходного дерева можно загрузить напрямую из JSON-документа с помощью команды **Загрузить из JSON...**

В открывшемся диалоговом окне доступны две вкладки для настройки параметров загрузки:

- На вкладке *Из файла* можно загрузить структуру из JSON-файла. Файл указывается в соответствующем поле с помощью диалогового окна «Открыть файл». Для выбора будут доступны только файлы с расширением .json.
- На вкладке *Из строки* предусмотрена возможность ручной вставки JSON-текста в многострочное поле.

**Шаблон даты/времени** — определяет формат распознавания строковых значений как дат. Его можно задать с помощью:

- Предустановленных шаблонов:
  - `YYYY-MM-DDThh:mm:ss` (ISO 8601, применяется по умолчанию)
  - `MM/DD/YYYY hh:mm:ss`
  - `DD.MM.YYYY hh:mm:ss`
- Вручную

Кнопка **Загрузить** в разделе **Управление загрузкой** запускает парсинг JSON и построение дерева, при этом существующая структура дерева будет перезаписана. В случае ошибок отобразится диагностическое сообщение. Кнопка **Отмена** закрывает диалог без сохранения изменений.

> **Важно:** При загрузке схемы существующие узлы будут удалены. Некорректный JSON или несоответствие шаблону даты приведет к ошибке загрузки.

# Особенности работы компонента

### Ограничение на разнородные массивы

Массивы должны содержать элементы одного типа или одинаковой структуры. Смешанные типы данных не поддерживаются.
**Пример:**

```
{
    "array": [12, "text", true, {}]  // Ошибка преобразования в дерево:
разные типы элементов в массиве
}
```

## Обработка дублирующихся ключей

При наличии в JSON нескольких узлов с одинаковыми именами в дерево включается только последний из дублирующихся узлов.
**Пример:**

```
{
    "color": "gold",
    "color": "зеленый" // В дерево попадёт только этот узел
}
```

Если дублирующиеся ключи имеют значения разных типов, результирующему узлу в дереве присваивается тип данных «переменный» и задается последнее значение ключа.
**Пример:**

```
{
    "name": "комментарий",
    "name": true
}

/* Результат */
Root
└── name   true  // Последнее значение узла, тип данных «переменный»
```

## Запрет «висящих» запятых

Запятая, стоящая после последнего элемента в объекте или массиве приводит к ошибке.
**Пример:**

```
{
    "array": [1, 2, 3,], // Ошибка: лишняя запятая в массиве
    "key": "value",      // Ошибка: лишняя запятая в объекте
}
```

## Обработка отсутствующих узлов

При ручной настройке структуры дерева или загрузке JSON-документа возможны два типа несоответствий:

- **Узел определен в дереве, но отсутствует в JSON** — примитивные узлы, отсутствующие в JSON, получают значение `null` в выходном дереве.

  **Пример:**

  ```
  /* Входной JSON */
  { "name": "Alice" }

  /* Структура дерева */
  Root
  ├── name  (тип: Строка)
  ├── age   (тип: Число)
  └── city  (тип: Строка)

  /* Результат */
  Root
  ├── name  Alice
  ├── age    null   // Поле age объявлено, но отсутствует в JSON
  └── city   null   // Поле city объявлено, но отсутствует в JSON
  ```

  *Контейнеры* и *Массивы* автоматически исключаются из выходного дерева, если в JSON отсутствует объект с соответствующим именем.

  **Пример:**

  ```
  /* Входной JSON */
  { "user": { "name": "Alice" } }

  /* Структура дерева */
  Root
  └── user  (тип: Контейнер)
      ├── name      (тип: Строка)
      ├── delivery  (тип: Массив)
      └── address   (тип: Контейнер)
          └── city  (тип: Строка)

  /* Результат */
  Root
  └── user
      └── name   Alice
  ```

- **Узел присутствует в JSON, но отсутствует в структуре дерева** — если функция *Определять структуру автоматически* выключена, при попытке выполнить узел возникнет ошибка.

## Дата/время с обозначением часового пояса

При выборе стандартного шаблона `YYYY-MM-DDThh:mm:ss` (ISO 8601) система автоматически применяет локальное время операционной системы (сервера или рабочей станции в случае настольной редакции), если указан часовой пояс `Z`.

**Пример:**

```
{
    "datetime_utc": "2024-12-31T23:59:59Z"
}

/* Результат */
Root
└── datetime_utc 01.01.2024, 02:59 // В окне Быстрого просмотра не
отображаются секунды и миллисекунды
```