
ORACLE®

JavaOne™

ORACLE

What's new in JSR 367 Java API for JSON Binding

Dmitry Kornilov
EclipseLink MOXy & SDO Team Lead
Oracle Czech

dmitry.kornilov@oracle.com

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

1. What is JSONB?
2. JSR Status & Progress
3. What is in the spec
4. The Implementation (with demo)
5. What's next
6. Q&A session

What is JSON Binding?

- JSON Binding is a standard
- JSON Binding = JSON-B = JSONB = JSR 367
- It’s about converting Java objects to and from JSON documents

What is JSON Binding?

Java

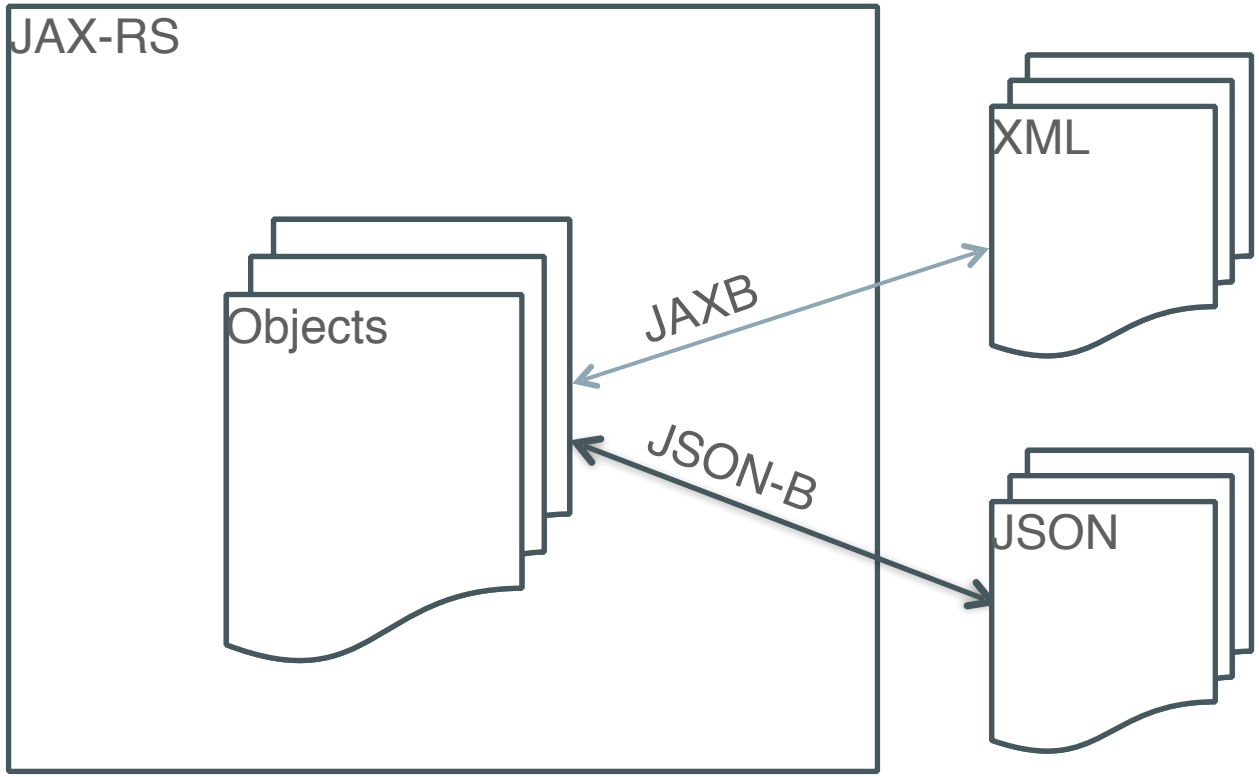
```
public class Customer {
    public int id;
    public String firstName;
    public String lastName;
   ….
}
```

```
Customer e = new Customer();
e.id = 1;
e.firstName = “John”;
e.lastName = “Doe”;
```

JSON

```
{
    "id": 1,
    "firstName" : "John",
    "lastName" : "Doe",
}
```

Usage in Other Frameworks



JSR Status & Progress

JSR 367 Status

https://www.jcp.org/en/jsr/detail?id=367

JSRs: Java Specification Requests

JSR 367: Java™ API for JSON Binding (JSON-B)

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Aug, 2015	19 Sep, 2015
Expert Group Formation		23 Sep, 2014	23 Jun, 2015
JSR Review Ballot	View results	09 Sep, 2014	22 Sep, 2014
JSR Review		26 Aug, 2014	08 Sep, 2014

Status: Active
JCP version in use: 2.9
Java Specification Participation Agreement version in use: 2.0

What's Done in Last Year

- Finished experts group formation
- Early Draft published
- Reference implementation started

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Aug, 2015	19 Sep, 2015
Expert Group Formation		23 Sep, 2014	23 Jun, 2015
JSR Review Ballot	View results	09 Sep, 2014	22 Sep, 2014
JSR Review		26 Aug, 2014	08 Sep, 2014

Experts Group

Team

Specification Leads

Martin Grebac	Oracle

Expert Group

Przemyslaw Bielicki	Eugen Cepoi	Datlowe : Martin Vojtek
IBM : Rick Curtis	IBM : Nathan Rauh	Oracle : Martin Grebac
Oracle : Dmitry Kornilov	Alexander Salvanos	Hendrik Saly
Santana, Otavio	Inderjeet Singh	TmaxSoft, Inc. : Kyung Koo Yoon
Tomitribe : Romain Manni-Bucau	Gregor Zurowski	

- I would like to join this Expert Group

Early Draft

- Created 20 Aug 2015
- Published 19 Sep 2015

Stage	Access	Start	Finish
Early Draft Review	Download page	20 Aug, 2015	19 Sep, 2015
Expert Group Formation		23 Sep, 2014	23 Jun, 2015
JSR Review Ballot	View results	09 Sep, 2014	22 Sep, 2014
JSR Review		26 Aug, 2014	08 Sep, 2014

Reference Implementation Started

http://git.eclipse.org/c/eclipselink/eclipselink.runtime.git/log/

```
author              Dmitry Kornilov     2015-09-14 09:06:06 (EDT)
committer           Dmitry Kornilov     2015-09-14 09:06:18 (EDT)
```

First JSONB marshaller implementation

```
Signed-off-by: Dmitry Kornilov <dmitry.kornilov@oracle.com>
Reviewed-by: Martin Grebac, Lukas Jungmann
```

Participate!

users@jsonb-spec.java.net

JSON-B Specification (Early Draft)

Specification Project

- Project Home: https://java.net/projects/jsonb-spec/pages/Home
- Sources & samples: https://java.net/projects/jsonb-spec/sources/git/show/api
- Document (pdf): https://java.net/projects/jsonb-spec/sources/git/content/spec/spec.pdf

JSON-B Runtime API

1. Create JSONB engine configuration (optional)
2. Create JSONB engine
3. Use it!

JSON-B Engine Configuration

- Configuration is optional
- Managed by one class
- Builder pattern is used
- Passed to **JsonbBuilder**

```
import javax.json.bind.JsonbConfig;

// JSON-B engine configuration
JsonbConfig config = new JsonbConfig()
    .withFormatting(…)
    .withNullValues(…)
    .withEncoding(…)
    .withStrictIJSON(…)
    .withPropertyNamingStrategy(…)
    .withPropertyOrderStrategy(…)
    .withPropertyVisibilityStrategy(…)
    .withAdapters()
    .withBinaryDataStrategy(…);
```

JSON-B Engine

- **Jsonb** class
- Created by **JsonbBuilder**
- Ability to choose JSONP provider

```
import javax.json.bind.Jsonb;
import javax.json.bind.JsonbBuilder;

// Create JSON-B engine
Jsonb jsonb = JsonbBuilder.newBuilder()
        .withConfig(…)
        .withProvider(…)
        .build();

// Create with given config
Jsonb jsonb = JsonbBuilder.create(config);

// Create with default config
Jsonb jsonb = JsonbBuilder.create();
```

JSON-B Engine

- toJson(…)
- fromJson(…)

```
String toJson(Object object) throws JsonbException;
String toJson(Object object, Type runtimeType) throws JsonbException;
void toJson(Object object, Appendable appendable) throws JsonbException;
void toJson(Object object, Type runtimeType, Appendable appendable)
     throws JsonbException;
void toJson(Object object, OutputStream stream) throws JsonbException;
void toJson(Object object, Type runtimeType, OutputStream stream)
     throws JsonbException;


<T> T fromJson(String str, Class<T> type) throws JsonbException;
<T> T fromJson(String str, Type runtimeType) throws JsonbException;
<T> T fromJson(Readable readable, Class<T> type) throws JsonbException;
<T> T fromJson(Readable readable, Type runtimeType) throws JsonbException;
<T> T fromJson(InputStream stream, Class<T> type) throws JsonbException;
<T> T fromJson(InputStream stream, Type runtimeType) throws JsonbException;
```

Default Mapping

- Basic Types
- Specific Types
- Dates
- Classes
- Collections/Arrays
- Enumerations
- JSON-P

```
import javax.json.bind.Jsonb;
import javax.json.bind.JsonbBuilder;

// Create with default config
Jsonb jsonb = JsonbBuilder.create();
```

Default Mapping - Basic Types

- java.lang.String
- java.lang.Character
- java.lang.Byte (byte)
- java.lang.Short (short)
- java.lang.Integer (int)
- java.lang.Long (long)
- java.lang.Float (float)
- java.lang.Double (double)
- java.lang.Boolean (boolean)
- java.lang.Number

```
Long longVal = Long.valueOf(1L);
String json = jsonb.toJson(longVal);

// Corresponding toString() method is used on serialization
json.equals(longVal.toString());  // true

// Corresponding parseX() method is used on deserialization
jsonb.fromJson(json, Long.class)
   .equals(Long.parseLong(json)); // true


jsonb.toJson("string");         // “string”
jsonb.toJson('\uFFFF');         // "\uFFFF\"
jsonb.toJson((byte)1);          // 1
jsonb.toJson((short)1);         // 1
jsonb.toJson((int)1);           // 1
jsonb.toJson(1L);               // 1
jsonb.toJson(1.2f);             // 1.2
jsonb.toJson(1.2);              // 1.2
jsonb.toJson(true);             // true
jsonb.toJson(null);             // null
```

Default Mapping - Specific Types

- java.math.BigInteger
- java.math.BigDecimal
- java.net.URL
- java.net.URI
- java.util.Optional
- java.util.OptionalInt
- java.util.OptionalLong
- java.util.OptionalDouble

```
BigDecimal bdVal = BigDecimal.valueOf(1.2);
String json = jsonb.toJson(bdVal);  // 1.2

// Corresponding toString() method is used on serialization
json.equals(bdVal.toString()); // true

// Constructor with string argument is used on deserialization
jsonb.fromJson(json, BigDecimal.class)
   .equals(new BigDecimal(1.2)); // true

// URL
URL url = new URL("http://jsonb-spec.java.net");
jsonb.toJson(url);      // "http://jsonb-spec.java.net"

// URI
URI uri = new URI("mailto:users@jsonb-spec.java.net");
jsonb.toJson(uri);      // "mailto:users@jsonb-spec.java.net"

// OptionalInt
jsonb.toJson(OptionalInt.of(1));    // 1
jsonb.toJson(OptionalInt.empty());  // null
```

Default Mapping - Dates

Java Type	Format
java.util.Date	ISO_DATE_TIME
java.util.Calendar	*ISO_DATE if to time information present, otherwise ISO_DATE_TIME*
java.util.GregorianCalendar	*ISO_DATE if to time information present, otherwise ISO_DATE_TIME*
java.util.TimeZone	NormalizedCustomId (see TimeZone javadoc)
java.util.SimpleTimeZone	NormalizedCustomId (see TimeZone javadoc)
java.time.Instant	ISO_INSTANT
java.time.LocalDate	ISO_LOCAL_DATE
java.time.LocalTime	ISO_LOCAL_TIME
java.time.LocalDateTime	ISO_LOCAL_DATE_TIME
java.time.ZonedDateTime	ISO_ZONED_DATE_TIME
java.time.OffsetDateTime	ISO_OFFSET_DATE_TIME
java.time.OffsetTime	ISO_OFFSET_TIME
java.time.ZoneId	NormalizedZoneId as specified in ZoneId javadoc
java.time.ZoneOffset	NormalizedZoneId as specified in ZoneOffset javadoc
java.time.Duration	ISO 8601 seconds based representation
java.time.Period	ISO 8601 period representation

Default Mapping - Date Samples

```
// java.util.Date
SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy");
Date parsedDate = sdf.parse("25.10.2015");
jsonb.toJson(parsedDate));                                  // ”2015-10-25T00:00:00"

// java.util.Calendar
Calendar dateCalendar = Calendar.getInstance();
dateCalendar.clear();
dateCalendar.set(2015, 10, 25);
jsonb.toJson(dateCalendar);                                 // ”2015-10-25”

// java.time.Instant
jsonb.toJson(Instant.parse(“2015-10-25T23:00:00Z"));        // ”2015-10-25T23:00:00Z”

// java.time.Duration
jsonb.toJson(Duration.ofHours(5).plusMinutes(4));           // “PT5H4M"

// java.time.Period
jsonb.toJson(Period.between(
    LocalDate.of(1960, Month.JANUARY, 1),
    LocalDate.of(1970, Month.JANUARY, 1)));                 // "P10Y"
```

Default Mapping - Classes

- Public and protected **nested** and **static nested** classes
- **Anonymous** classes (serialization only)
- Default no-argument constructor is required for deserialization

Default Mapping - Fields

- Fields serialized in **lexicographical** order
- Parent class fields are serialized before child class fields
- **Final** fields are supported (serialization only)
- **Static** fields are not supported
- **Transient** fields are not supported
- **Null** fields are skipped

Fields Order Sample

```
public class Parent {
    public int parentB;
    public int parentA;
}

public class Child extends Parent {
    public int childB;
    public int childA;
}
```

```
{
  "parentA": 1,
  "parentB": 2
}

{
  "parentA": 1,
  "parentB": 2,
  "childA": 3,
  "childB": 4
}
```

Default Mapping - Scope and Field Access Strategy

Serialization

- Existing fields with public getters
- Public fields with no getters
- Public getter/setter pair without a corresponding field

Deserialization

- Existing fields with public setters
- Public fields with no setters
- Public getter/setter pair without a corresponding field

Scope and Field Access Strategy Sample

```
public class Foo {
    public final int publicFinalField;
    private final int privateFinalField;

    public static int publicStaticField;

    public int publicFieldWithNoGetter;
    public int publicFieldWithPrivateGetter;
    public Integer publicNullField = null;

    private int privateFieldWithNoGetter;
    private int privateFieldWithPublicGetter;

    public int getNoField() {};
    public void setNoField(int value) {};
}
```

```
{
  "publicFinalField": 1,

  "publicFieldWithNoGetter": 1,

  "privateFieldWithPublicGetter": 1,

  "noField": 1,
}
```

Default Mapping - Arrays/Collections

- Collection
- Map
- Set
- HashSet
- NavigableSet
- SortedSet
- TreeSet
- LinkedHashSet
- TreeHashSet
- HashMap
- NavigableMap
- SortedMap
- TreeMap
- LinkedHashMap
- TreeHashMap
- List
- ArrayList
- LinkedList
- Deque
- ArrayDeque
- Queue
- PriorityQueue
- EnumSet
- EnumMap

```
// Array
int[] intArray = {1, 2, 3};

jsonb.toJson(intArray);       // [1,2,3]

// Collection
Collection<Object> list = new ArrayList<>();
list.add(1);
list.add(2);
list.add(null);

jsonb.toJson(list);           // [1,2,null]

// Map
Map<String, Object> map = new LinkedHashMap<>();
map.put("first", 1);
map.put("second", 2);

jsonb.toJson(map);            // {"first":1,"second":2}
```

Default Mapping - Enumerations

- Serialization:
  name()
- Deserialization:
  valueOf(String)

```
public enum Language {
    English, Russian, Czech
}

jsonb.toJson(Language.English);   // "English"
```

Default Mapping - JSON-P

- Supported types:
  - javax.json.JsonArray
  - javax.json.JsonStructure
  - javax.json.JsonValue
  - javax.json.JsonPointer
  - javax.json.JsonString
  - javax.json.JsonNumber
- Serialization must produce the same result as with **javax.json.JsonWriter**
- Deserialization must produce the same result as with **javax.json.JsonReader**

```
// JsonObject
JsonBuilderFactory factory = Json.createBuilderFactory(null);
JsonObject jsonObject = factory.createObjectBuilder()
        .add("name", "Jason")
        .add("city", "Prague")
        .build();

jsonb.toJson(jsonObject); // {"name":"Jason","city":"Prague"}

// JsonValue
jsonb.toJson(JsonValue.TRUE); // true
```

Reference Implementation Demo

Jason’s Plans

- Web: http://159.203.248.103:8080
- Sources: https://github.com/m0mus/jason_plans
- Demonstrates serialization of this class:

```
public class JasonPlans {
    public Date date;
    public int id;
    public String response;
}
```

Custom Mapping

- Property names
- Property order
- Ignoring properties
- Null handling
- Simple values

- Custom instantiation
- Custom visibility
- Adapters
- Date/Number Formats
- Binary Handling

Custom Mapping - Property Names

```java
public class Customer {
    public int id;
    public String firstName;
}
```

```json
{
  "id": 1,
  "firstName": "Jason"
}
```

Custom Mapping - Property Names

```
public class Customer {
    private int id;

    @JsonbProperty("name")
    private String firstName;
}
```

```
public class Customer {
    public int id;
    public String firstName;

    @JsonbProperty("name")
    public String getFirstName() {
        return firstName;
    }
}
```

```
{
  "id": 1,
  "name": "Jason"
}
```

Custom Mapping - Property Names

```
public class Customer {
    public int id;
    public String firstName;

    @JsonbProperty(“getter-name")
    String getFirstName() {
        return firstName;
    }

    @JsonbProperty(“setter-name")
    void setFirstName(String str) {
        this.firstName = str;
    }
}
```

Serialization:

```
{
  "id": 1,
  “getter-name": "Jason"
}
```

Deserialization:

```
{
  "id": 1,
  “setter-name": "Jason"
}
```

Custom Mapping - Property Naming Strategy

• IDENTITY	myMixedCaseProperty
• LOWER_CASE_WITH_DASHES	my-mixed-case-property
• LOWER_CASE_WITH_UNDERSCORES	my_mixed_case_property
• UPPER_CAMEL_CASE	MyMixedCaseProperty
• UPPER_CAMEL_CASE_WITH_SPACES	My Mixed Case Property
• CASE_INSENSITIVE	myMixedCaseProperty mYmIxEdCaSePrOpErTy

Custom Mapping - Property Naming Strategy

• IDENTITY	myMixedCaseProperty
• LOWER_CASE_WITH_DASHES	my-mixed-case-property
• LOWER_CASE_WITH_UNDERSCORES	my_mixed_case_property
• UPPER_CAMEL_CASE	MyMixedCaseProperty
• UPPER_CAMEL_CASE_WITH_SPACES	My Mixed Case Property
• CASE_INSENSITIVE	myMixedCaseProperty mYmIxEdCaSePrOpErTy

```
JsonbConfig config = new JsonbConfig()
   .withPropertyNamingStrategy(PropertyNamingStrategy.LOWER_CASE_WITH_DASHES);

Jsonb jsonb = JsonbBuilder.create(config);
String json = jsonb.toJson(obj);
```

Custom Mapping - Property Order Strategy

- Strategies:
  - LEXICOGRAPHICAL
  - ANY
  - REVERSE

```
{ "a": 1, "b": 2, "c": 3 }
{ "b": 2, "a": 1, "c": 3 }
{ "c": 3, "b": 2, "a": 1 }
```

- @JsonbPropertyOrder annotation on class
- JsonbConfig().withPropertyOrderStrategy(…)

Custom Mapping - Ignoring Properties

```
public class Customer {
    public int id;
    public String name;
}
```

```
{
  "id": 1,
  "name": "Jason"
}
```

```
public class Customer {
    public int id;

    @JsonbTransient
    public String name;
}
```

```
{
  "id": 1,
}
```

Custom Mapping - Null handling

```
public class Customer {
    private int id = 1;
    private String name = null;
}
```

```
{
  "id": 1
}
```

```
public class Customer {
    private int id = 1;

    @JsonbNillable
    private String name = null;
}
```

```
{
  "id": 1,
  "name": null
}
```

```
@JsonbNillable
public class Customer {
    private int id = 1;
    private String name = null;
}
```

```
{
  "id": 1,
  "name": null
}
```

Custom Mapping - Simple Value

```
public class Language {
    public int id = 1;

    @JsonbValue
    public String code = “en";

    public String name = "English";
}

public class Customer {
    public int id;
    public Language lang;
    public String name;
}
```

```
”en”
```

```
{
  "id": 1,
  "lang": ”en”,
  "name": ”Jason”
}
```

Custom Mapping - Custom Instantiation

```
public class Customer {
    public int id;
    public String name;

    @JsonbCreator
    public static Customer getFromDb(int id) {
        return CustomerDao.getByPrimaryKey(id);
    }
}

public class Order {
    public int id;
    public Customer cust;
}
```

```
{
  "id": 1,
  "cust": {
    "id": 1
  }
}
```

Custom Mapping - Custom Visibility

```
public class Customer {
    public int id;
    private String name;
}
```

```
{
  "id": 1
}
```

```
@JsonbVisibility(MyVisibilityStrategy.class)
public class Customer {
    public int id;
    private String name;
}
```

```
{
  "id": 1,
  "name": "Jason"
}
```

```
JsonbConfig config = new JsonbConfig()
   .withPropertyVisibilityStrategy(new MyVisibilityStrategy());
Jsonb jsonb = JsonbBuilder.create(config);
String json = jsonb.toJson(obj);
```

Custom Mapping - Adapters

```
public class Customer {
    public int id;
    public String name;

    @JsonbTypeAdapter(MyAdapter.class)
    public CustomerData data;
}
```

```
{
  "id": 1,
  "name": "Jason",
  "data": {…},
}
```

```
JsonbConfig config = new JsonbConfig().withAdapters(new MyAdapter());
Jsonb jsonb = JsonbBuilder.create(config);
String json = jsonb.toJson(obj);
```

Custom Mapping - Date/Number Format

```
public class FormatTest {
    public Date defaultDate;

    @JsonbDateFormat("dd.MM.yyyy")
    public Date formattedDate;

    public BigDecimal defaultNumber;

    @JsonbNumberFormat(“#0.00")
    public BigDecimal formattedNumber;
}
```

```
{
  “defaultDate”: “2015-07-26T23:00:00",

  “formattedDate”: ”26.07.2015",

  “defaultNumber": 1.2,

  “formattedNumber": 1.20
}
```

Custom Mapping - Binary Data Encoding

- BYTE (default)
- BASE_64
- BASE_64_URL

```
JsonbConfig config = new JsonbConfig()
   .withBinaryDataStrategy(BinaryDataStrategy.BASE_64);

Jsonb jsonb = JsonbBuilder.create(config);
String json = jsonb.toJson(obj);
```

Custom Mapping - I-JSON

- I-JSON (”Internet JSON”) is a restricted profile of JSON
- JSON-B fully supports I-JSON by default with three exceptions
- Configuration: withStrictIJSONSerializationCompliance

What’s Next

In Progress

- Early Draft 2
- Progress at jcp.org, jsonb-spec.java.net
- Reference implementation started at http://eclipselink.org

TODO

- Mapping subset of documents (JsonPointer)
- Mapping 3rd party objects
- Bi-directional mapping

Special Thanks

- Martin Grebac, Martin Vojtek and all the experts
- Kiev Java User Group (Olena Syrota, Oleg Tsal-Tsalko)
- David Delabassee, Reza Rahman, Heather VanCura, John Clingan
- others on users@jsonb-spec.java.net

Links

- Spec Project: https://java.net/projects/jsonb-spec/pages/Home
- JCP Home: https://www.jcp.org/en/jsr/detail?id=367
- Reference Implementation: http://eclipselink.org

Call to Action!

- Participate!
  users@jsonb-spec.java.net
- Contribute!
  http://eclipselink.org

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Integrated Cloud

Applications & Platform Services

JavaOne™

ORACLE


ORACLE®