

# Oracle8<sup>TM</sup> Visual Information Retrieval Java Client

User's Guide and Reference

Release 8.1.5

February 1999

Part No. A67300-01

Visual Information Retrieval technology licensed  
from Virage, Inc.

---

Oracle8i Visual Information Retrieval Java Client User's Guide and Reference

Part No. A67300-01

Release 8.1.5

Copyright © 1999, Oracle Corporation. All rights reserved.

Primary Author: Jeff Hebert

Contributing Author: Max Chittister

Contributors: Donna Brown, Sam Lee, Sue Mavris, Dan Mullen, and Deb Laderoute.

**The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the Programs.**

This Program contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright, patent and other intellectual property law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation

If this Program is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

**Restricted Rights Legend** Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication and disclosure of the Programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data -- General, including Alternate III (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle is a registered trademark, and Oracle8, Oracle8i, Oracle Call Interface, PL/SQL, and Network Computing Architecture are trademarks of Oracle Corporation, Redwood City, California.

Virage is a registered trademark of Virage, Inc.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

---

---

# Contents

<b>Send Us Your Comments .....</b>	<b>v</b>
<b>Preface.....</b>	<b>vii</b>
Intended Audience .....	vii
Structure.....	vii
Related Documents.....	vii
Conventions.....	viii
<b>1 Introduction</b>	
1.1 Relationship with Oracle8i <i>interMedia</i> Image.....	1-1
1.2 Client-Side Support.....	1-2
1.3 Client/Server Similarities and Differences.....	1-3
1.4 Client/Server Interaction .....	1-3
<b>2 OrdVir Class Library</b>	
2.1 OrdVir Object Type.....	2-1
2.2 Methods .....	2-4
2.2.1 Inherited Methods.....	2-5
2.2.2 OrdVir Methods for Visual Information Retrieval.....	2-8
analyze Method .....	2-9
convert Method.....	2-10
convert Method (static).....	2-12
score Method.....	2-14

	score Method (static) .....	2-16
	similar Method.....	2-19
	similar Method (static).....	2-22
	getSignature Method.....	2-25
	setSignature Method .....	2-26
2.2.3	OrdVir Methods for Communication Between the Client and Server .....	2-27
	flush() Method.....	2-28
	refresh() Method .....	2-29

## **Glossary**

## **Index**

---

---

# Send Us Your Comments

**Oracle8i Visual Information Retrieval Java Client User's Guide and Reference, Release 8.1.5**  
**Part No. A67300-01**

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail: [nedc\\_doc@us.oracle.com](mailto:nedc_doc@us.oracle.com)
- FAX - 603.897.3316. Attn: Visual Information Retrieval Documentation
- Postal service:  
Oracle Corporation  
Visual Information Retrieval Documentation  
One Oracle Drive  
Nashua, NH 03062  
USA

If you would like a reply, please give your name, and a postal or e-mail address.

If you have problems with the software, please contact your local Oracle Worldwide Support Center.



---

# Preface

## Intended Audience

This guide is intended for anyone who is interested in storing, retrieving, and manipulating image data in an Oracle8i database using Oracle8i Visual Information Retrieval and a Java client interface, including developers of image specialization services.

## Structure

This guide contains the following chapters:

- [Chapter 1](#) Introduces Visual Information Retrieval and explains image-related concepts.
- [Chapter 2](#) Provides reference information about the Visual Information Retrieval Java class methods.
- [Glossary](#) Defines important terms related to data services and Visual Information Retrieval.

## Related Documents

This guide is not intended as a standalone document. It is a supplement to *Oracle8i Visual Information Retrieval User's Guide and Reference*, specifically, to describe the Java client interface. You need both guides to successfully perform visual information retrieval using the Java interface to this product.

In addition, the OrdVir Java class library described in this guide contains many inherited methods that are documented in *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and*

*Reference.* This guide should also be considered essential in understanding and using the Java interface to this product.

---

---

**Note:** For information added after the release of this guide, refer to the online README.txt file under your *ORACLE\_HOME* directory. Depending on your operating system, this file may be in:

*ORACLE\_HOME/ord/vir/admin/README.txt*

Please see your operating-system specific installation guide for more information.

---

---

For more information about using this product in a development environment, see the following documents in the Oracle8i documentation set:

- *Oracle Call Interface Programmer's Guide*
- *Oracle8i Application Developer's Guide - Large Objects (LOBs)*
- *Oracle8i Concepts*
- *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference*
- *PL/SQL User's Guide and Reference*

For information about basic image storage and retrieval, see *Oracle8i interMedia Audio, Image, and Video User's Guide and Reference*.

Visual Information Retrieval is based on technology licensed from Virage, Inc. Visit the Virage Web site for additional information about this ongoing relationship:  
<http://www.virage.com/oracle>.

## Conventions

In this guide, Oracle8i Visual Information Retrieval is sometimes referred to as simply, “the product” or “the option”.

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this guide:

---

<b>Convention</b>	<b>Meaning</b>
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
...	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
<b>boldface text</b>	Boldface text indicates a term defined in the text, the glossary, or in both locations.
<i>italic text</i>	Italic text is used for emphasis, for user-supplied variables, and for book titles.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

---



---

# Introduction

Oracle8i Visual Information Retrieval is an extension to Oracle8i Enterprise Edition that provides image storage, content-based retrieval, and format conversion capabilities through an object type. The capabilities of this product encompass the storage, retrieval, and manipulation of image data managed by the Oracle8i Enterprise Edition database server. This product supports image storage using binary large objects (BLOBs) and references to image data residing externally in binary files (BFILEs) or URLs.

Visual Information Retrieval is a building block for various imaging applications, rather than being an end-user application in itself. It consists of an object type (OrdVir) along with related methods for managing and processing image data.

Refer to *Oracle8i Visual Information Retrieval User's Guide and Reference* for information about how visual information retrieval works.

---

**Note:** Many of the methods mentioned in this chapter are described in the *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference*. This guide describes only methods that are unique or changed from the superclass definition.

---

## 1.1 Relationship with Oracle8i *interMedia* Image

You may already be familiar with Oracle8i *interMedia* Image, either as a component of Oracle8i *interMedia* or in a previous release as Oracle8 Image Cartridge. The base Image option and Visual Information Retrieval both let you store an image as an object in the database or as a reference to an external file or URL. Both products let you store and query on the following attributes:

- Image height

- Image width
- Image size
- File type or format (such as TIFF)
- Compression type or format (such as JPEG)
- Image type (such as monochrome)
- MIME type

Understanding the base Image option is important because the Visual Information Retrieval object type is defined as the Image object plus a signature attribute.

The main differences between the Image option and the Visual Information Retrieval product are that Visual Information Retrieval lets you create and use indexes, and perform **content-based retrieval**. Content-based retrieval lets you perform queries based on intrinsic visual attributes of the image (color, structure, texture), rather than being limited to keyword searches in textual annotations or descriptions. The underlying technology was developed by Virage, Inc., a leader in content-based retrieval.

Visual Information Retrieval can also be used for facial image recognition when used in conjunction with software from Viisage Technology, Inc. Facial image recognition has applications ranging from security, by determining if the holder of an ATM card is the owner of the card, to movie casting, by determining which actor looks most like a particular historical figure.

## 1.2 Client-Side Support

Visual Information Retrieval lets you store image data in a database table. The Java client aspect of the product lets you develop client-side Java applications to manipulate and/or modify image data stored in a network-accessible (server-side) database.

You can connect to a server-side object, copy that object from the server side to the client side, perform various operations on the client-side object, and transfer the new object back to the server side.

For situations where you do not have permission to modify the server-side object, Visual Information Retrieval Java Client can retrieve the image from the server side for display purposes only.

## 1.3 Client/Server Similarities and Differences

Client-side objects have many similarities to server-side objects of the same type.

For each client-side object, there must be a corresponding server-side object. The objects are identical, and the client-side object contains all the APIs that the server-side object does.

However, there are some architectural differences between the two. Visual Information Retrieval Java Client makes use of Java inheritance to build the `OrdVir` class from the `OrdImage` and `OrdMultiMedia` superclasses.

There are a number of attributes that are included on the client side that are not on the server side. These include the following:

- A connection attribute that connects to the server side.
- Binding attributes that are used to bind to a specific server-side object.
- A lock attribute that selects the server-side object for update so changes on the client side will also be made on the server-side object.

The client side also includes some APIs that are not included on the server side. These APIs are used to do the following:

- Connect to the server-side object (`setConnection()`).
- Bind to the server-side object (`setBindParams()`).
- Update the client-side object from the server-side object (`refresh()`).
- Update the server-side object from the client-side object (`flush()`).
- Lock the server-side object so it can be modified (`setLock()`).

## 1.4 Client/Server Interaction

The client/server interaction described in this section assumes that you have created an `OrdVir` object on the client side.

The following steps describe the process of modifying an `OrdVir` object using a client/server model.

1. Make a connection from the client to the server with the `setConnection()` method.
2. Bind a client-side object to a matching server-side object based on the `tableName`, `columnName`, and condition variables.

3. Refresh the client-side object. The server sends information about the server-side object to the client-side object. The client-side object is then updated to exactly match the server-side object.

It is possible to refresh in two different modes: for read-only purposes and for updating purposes. The difference between the two is the lock value. If the lock value is *false*, then you do not have permission to make changes to the object. If the lock value is *true*, then you have permission to make changes to the object.

4. Perform *some* server-side image processing operations from the client-side object. Note that if you execute a method that operates on the server-side object and it changes the state of the server-side object, you must explicitly refresh the client-side object to reflect that change.
5. Flush the client-side object. Once you finish using the client-side object, the client sends information about the client-side object to the server-side object. The server-side object is then updated to exactly match the client-side object.

Note that if you execute a method that operates on the client-side object and it changes the state of only the client-side object, you must explicitly flush the server-side object to reflect that change on the server side.

6. If you set a lock, it is not automatically released upon flushing. You must release the lock manually using the `setLock()` method.

If [Table 1-1](#) indicates that a method operates on both sides, this means that the method operates on the server side, but afterward, it synchronizes with the attributes on the client side.

**Table 1-1 Client/Server Interactions for *OrdVir* and Superclass Methods**

Method	OrdVir Object Location
<code>analyze</code>	Client, Server
<code>checkProperties</code>	Server
<code>clearLocal</code>	Client
<code>convert</code>	Server
<code>convert (static)</code>	Server
<code>copy</code>	Server
<code>deleteContent</code>	Server
<code>getAllAttributesAsString</code>	Client
<code>getBFILE</code>	Server

**Table 1–1 Client/Server Interactions for OrdVir and Superclass Methods (Cont.)**

getCompressionFormat	Client
getContent	Client, Server
getContentFormat	Client
getContentLength	Client
getData (2 methods)	Server
getDataInFile	Server
getDataInStream	Server
getFormat	Client
getHeight	Client
getMimeType	Client
getSignature	Client
getSource	Client
getSourceLocation	Client
getSourceName	Client
getSourceObject	Client
getSourceType	Client
getUpdateTime	Client
getWidth	Client
importData	Server
importFrom	Server
isLocal	Client
loadData (2 methods)	Server
process	Server
processCopy	Server
score	Operates on the server side against the client-side signature
score (static)	Operates on the server side against the client-side signature
setFormat	Client

**Table 1–1 Client/Server Interactions for OrdVir and Superclass Methods (Cont.)**

---

setLocal	Client
setMimeType	Client
setProperties	Client, Server
setProperties(String)	Client, Server
setSignature	Client
setSource	Client
setSourceInformation	Client
setUpdateTime	Client, Server
similar	Operates on the server side against the client-side signature
similar (static)	Operates on the server side against the client-side signature

---

---

# OrdVir Class Library

The Oracle8i Visual Information Retrieval Java Client class library contains information about the OrdVir object type:

- Object type -- see [Section 2.1](#)
- Methods -- see [Section 2.2](#)

The OrdVir object type and class library are based on the OrdImage object type, which are, in turn, based on the OrdMultiMedia object type. OrdMultiMedia contains the OrdSource object type. This guide describes only information unique to the Visual Information Retrieval product. Any inherited methods or other characteristics are discussed in their own documentation.

The *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference* describes the OrdImage, OrdMultiMedia, and OrdSource object types.

## 2.1 OrdVir Object Type

The OrdVir object type supports the storage and management of image data.

---

---

**Note:** In the following code, some methods will begin with "public". This indicates that the method can be called by all classes in all packages. Methods that begin with "protected" are visible to all classes in the package, plus all subclasses.

---

---

This object type is defined by Oracle8i Visual Information Retrieval as follows:

```
package oracle.ord.media;  
import java.sql.*;  
import java.io.*;  
import oracle.jdbc.driver.*;
```

```
import oracle.sql.*;
import oracle.ord.media.*;

public class OrdVir extends OrdImage {

    // OrdVir class attribute
    byte[] signature;

    // OrdVir Default Constructor
    public OrdVir() {}

    public OrdVir(Connection connection)
    { }

    int bindInSQLParams(int start, OracleCallableStatement stmt)
    throws SQLException
    { }

    String defineSQLResults(String var)
    { }

    int declareSQLResults(int start, OracleCallableStatement stmt)
    throws SQLException
    { }

    String setSQLParams(String var)
    { }

    int getSQLResults(int start, OracleCallableStatement stmt)
    throws SQLException
    { }

    String getMediaType()
    { }

    String getFormatStr()
    { }

    String getUpdStr()
    { }
```

```
String getSourceStr()
{ }

String getContentLengthAPI()
{ }

public void refresh(boolean forUpdate) throws SQLException
{ }

public void flush() throws SQLException
{ }

public void analyze() throws SQLException
{ }

public int similar(byte [] signature2, String attrWeights, float threshold)
throws SQLException
{ }

public float score(byte [] signature2, String attrWeights)
throws SQLException
{ }

public int convert(String operations)
throws SQLException
{ }

// Vir Operators:
public static int similar(byte [] signature1, byte [] signature2,
                        String attrWeights, float threshold,
                        Connection connection)
throws SQLException
{ }

public static float score(byte [] signature1, byte [] signature2,
                        String attrWeights, Connection connection)
throws SQLException
{ }

public static int convert(byte [] signature, String operations,
                        Connection connection)
throws SQLException
{ }
```

```
// Accessor functions for local OrdVir attributes
public byte[] getSignature() throws SQLException
{ }

public void setSignature(byte[] value) throws SQLException
{ }

}
```

Where the base image attributes included from OrdImage.java are defined as follows:

- **height**: contains the height of the image in pixels.
- **width**: contains the width of the image in pixels.
- **contentLength**: contains the size of the image data in bytes.
- **contentType**: describes the type of image (for example, monochrome or 8-bit grayscale).
- **compressionFormat**: describes the type of compression used on the image.

Three additional attributes of special note are defined in the included files, oracle.ord.media.\*:

- **source**: contains the image in the ORDSource object.
- **fileFormat**: describes the file type or format in which the image data is stored (TIFF, JIFF, and so forth).
- **mimeType**: describes the MIME type of the image.

See *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference* for information about the oracle.ord.media.\* files and the OrdMultiMedia abstract class.

## 2.2 Methods

This section presents OrdVir reference information on the methods used for image data manipulation.

The OrdVir methods are described in the following groupings:

### **OrdVir Methods for Visual Information Retrieval**

- `public void analyze()`: generates a signature based on the visual attributes of an image.
- `public int convert()`: converts the signature to the format used on the host system.
- `public static convert()`: converts the signature to the format used on the host system, without requiring an instance of an OrdVir object.
- `public float score()`: compares the signatures of two images, returning a value between 0.0 (perfect match) and 100 (totally opposite).
- `public static float score()`: compares the signatures of two images, returning a value between 0.0 and 100, without requiring an instance of an OrdVir object.
- `public int similar()`: compares the current image to a provided comparison signature and determines if they are alike within a specified threshold.
- `public static int similar()`: compares the signatures of two images and determines if they are alike within a specified threshold, without requiring an instance of an OrdVir object.
- `public getSignature()`: reads the signature of the OrdVir object stored on the client side.
- `public setSignature()`: sets the signature field of the OrdVir object.

### **OrdVir Methods for Communication Between the Client and Server**

- `public void flush()`: sends information from the client-side OrdVir object to the server-side OrdVir object.
- `public void refresh()`: sends information from the server-side OrdVir object to the client-side OrdVir object with or without locking the database row.

## **2.2.1 Inherited Methods**

The following methods are inherited through the OrdMultiMedia package. See *Oracle8i interMedia Audio, Image, and Video Java Client User's Guide and Reference* for further information on these methods.

### **OrdVir Methods Associated with Image Attributes**

- `public int getHeight()`: gets the image height from the client cache.
- `public int getWidth()`: gets the image width from the client cache.

- `public int getContentLength():` gets the length of the content from the client cache.
- `public String getContentType():` gets the content format from the client cache.
- `public String getCompressionFormat():` gets the compression format from the client cache.
- `public String getAllAttributesAsString():` gets all the image attributes and returns them as a string.

#### **OrdVir Methods Associated with the Source Attribute**

- `public BLOB getContent():` gets the content BLOB that holds the image data.
- `public BFILE getBFILE():` gets the content BFILE that holds the image data.
- `public void deleteContent():` deletes the image data from the LOB on the server-side OrdVir object.
- `public String getSource():` returns information about the data source of the OrdVir object.

#### **OrdVir Methods Associated with Properties Set and Check Operations**

- `public void setProperties():` sets the database object attributes according to the values stored in the content data header.
- `public void setProperties(String):` sets the database object attributes according to the values provided in the parameter.
- `public boolean checkProperties():` checks that the values of the database OrdVir object are consistent with the values stored in the content header.

#### **OrdVir Methods Associated with Copy Operations**

- `public void copy():` copies the image data from the client-side OrdVir object source to a server-side OrdVir object.

#### **OrdVir Methods Associated with Processing Image Data**

- `public void process():` executes a given command on the server side OrdVir object and updates that object.
- `public void processCopy():` copies image data to a destination OrdVir object and modifies the copy according to the specified command.

---

### **OrdVir Methods Associated with the Media Type**

- protected String getMediaType(): returns the media type information.

### **OrdVir Methods Associated with Locking**

- protected void setLock(): sets the lock on the OrdVir object.

### **OrdVir Methods Associated with the SQL Type**

- protected int bindInSQLParams(): binds the local values of the OrdVir object attributes to the parameters of the given statement.
- protected String defineSQLResults(): constructs a SQL statement to transfer data from a fetched object to the local object attributes.
- protected int declareSQLResults(): binds the output parameters in a statement produced by the defineSQLResults() method to the attributes of the OrdVir object.
- protected String setSQLParams(): constructs a SQL statement to set the attributes of a database OrdVir object to the values specified by the binding parameters.
- protected int getSQLResults(): updates the attributes of the client-side OrdVir object with values from the database object fetched by the executing statement.

### **OrdVir Methods Associated with Generating SQL Queries**

- protected String getFormatStr(): returns the format string.
- protected String getUpdStr(): returns the SQL string that is used to update an OrdVir instance.
- protected String getSourceStr(): returns the source string.
- protected String getContentLengthAPI(): returns the server-side contentLength API for this class.

## 2.2.2 OrdVir Methods for Visual Information Retrieval

This section presents reference information on the OrdVir methods associated directly with visual information retrieval.

The methods described in this chapter show examples based on the instantiation of an OrdVir object. For the examples in this section, please consult the following Java code:

```
import oracle.ord.media.mediaClass;

public class clientProgram {
    public static void main(String[] args){

        The code in the examples should be placed here.
    }
}
```

In each example, the parameter connection is a connection to an Oracle8i database.

---

## analyze Method

### Format

```
public void analyze( )  
throws SQLException
```

### Scope

public

### Description

Sets the value for both the server- and client-side signature attributes based on the image data.

### Parameter

None.

### Returns

None.

### Exception

java.sql.SQLException

### Usage

```
try{  
    OrdVir imageObj = new OrdVir (connection);  
    .  
    .  
    .  
    imageObj.analyze();  
}  
catch(SQLException e){  
    .  
    .  
    .  
}
```

---

## convert Method

### Format

```
public int convert(string operations)
throws SQLException
```

### Scope

public

### Description

Converts the server-side signature data to a format usable on the database server (host) machine.

### Parameter

#### **operations**

The type of processing done to the image signature. The following operations are available:

Operation Keyword	Description
BYTEORDER	Converts the signature to the natural byte order of the host machine.
VIISAGE	Converts the signature from the format used for Viisage facial image recognition to a signature usable by the score() and similar() operators.

### Returns

None.

### Exception

java.sql.SQLException

### Usage

When the operation is BYTEORDER, the signature is converted to the format of the host machine regardless of its initial state.

This method is useful if your signature data was created on a system with a different byte order than the system to which you are currently bound (the system where you intend to perform `similar()` and `score()` operations). If your host machine is from Sun Microsystems, Inc., the `Convert()` operator sets the signature to the big endian byte order. On an Intel Corporation machine, the operator converts the signature to the little endian byte order. Note that the images themselves are system-independent; only the signatures need to be converted.

When the operator is `VIISAGE`, the signature is converted from the format used by Viisage Technology for facial image recognition to the format usable by this product for `score` and similar operations.

```
try{
    OrdVir imageObj = new OrdVir (connection);
    .
    .
    .
    ImageObj.convert("BYTEORDER");
}
catch(SQLException e){
    .
    .
    .
}
```

---

## convert Method (static)

### Format

```
public static int convert(byte [] signature,  
                          string operations,  
                          connection connection )  
  
throws SQLException
```

### Scope

public

### Description

Converts the client-side signature data to a format usable on the host machine. This is a static function not within the context of a particular image object.

Unlike the non-static version of this function, the static function can directly operate on signature data without any instance of an OrdVir object.

### Parameters

#### **signature**

The signature of the image, as created by the analyze() method or by Viisage software. Data type is RAW(2000).

#### **operation**

The type of processing done to the image signature. The following operations are available:

<b>Operation Keyword</b>	<b>Description</b>
BYTEORDER	Converts the signature to the natural byte order of the host machine.
VIISAGE	Converts the signature from the format used for Viisage facial image recognition to a signature usable by the score() and similar() operators.

#### **connection**

An object representing a connection to the server.

## Returns

None.

## Exception

java.sql.SQLException

## Usage

When the operation is `BYTEORDER`, the signature is converted to the format of the host machine regardless of its initial state.

If your host machine is from Sun Microsystems, Inc., the `Convert()` operator sets the signature to the big endian byte order. On an Intel Corporation machine, the operator converts the signature to the little endian byte order. Note that the images themselves are system-independent; only the signatures need to be converted.

When the operator is `VIISAGE`, the signature is converted from the format used by Viisage Technology for facial image recognition to the format usable by this product for `score()` and `similar()` operations.

```
try{
    byte[] imgSignature = new byte[2000];
    .
    .
    .
    OrdVir.convert(imgSignature, "BYTEORDER", connection);
}
catch(SQLException e){
    .
    .
    .
}
```

## score Method

### Format

```
public float score(byte [] cmpSignature,  
                  string attrWeights)  
throws SQLException
```

### Scope

public

### Description

Compares the client-side image signature of the current OrdVir object to the cmpSignature signature using the weights provided in attrWeights.

### Parameters

#### **cmpSignature**

The signature of the comparison image.

#### **attrWeights**

A list of weights to apply to each visual attribute. Data type is VARCHAR2. The following attributes can be specified, with a value of 0.0 specifying no importance and a value of 1.0 specifying highest importance. You must specify a value greater than zero for at least one of the attributes. The facial attribute is not compatible with any other attributes.

Attribute	Description
globalcolor	The weight value (0.0 to 1.0) assigned to the global color visual attribute. Data type is NUMBER. Default is 0.0.
localcolor	The weight value (0.0 to 1.0) assigned to the local color visual attribute. Data type is NUMBER. Default is 0.0.

Attribute	Description
texture	The weight value (0.0 to 1.0) assigned to the texture visual attribute. Data type is NUMBER. Default is 0.0.
structure	The weight value (0.0 to 1.0) assigned to the structure visual attribute. Data type is NUMBER. Default is 0.0.
facial	A value of 1 indicates that this is a facial signature. Data type is NUMBER. Default is 0.0.

---



---

**Note:** When specifying parameter values that include floating-point numbers, you should use double quotation marks ( " ") around the value. If you do not, this may result in incorrect values being passed, and you will get incorrect results.

---



---

## Returns

A floating-point number between 0.0 and 100.0, representing the weighted sum of the distances between the visual attributes.

## Exception

java.sql.SQLException

## Usage

The connection string of the two image objects must be the same.

```
try{
    byte signature2 = new byte[2000];
    .
    .
    .
    float score= OrdVir.score(signature2, "localcolor=1.0",connection);
}
catch(SQLException e){
    .
    .
    . }
}
```

## score Method (static)

### Format

```
public static float score(byte [] signature1,  
                          byte [] signature2,  
                          string attrWeights  
                          connection connection)  
  
throws SQLException
```

### Scope

public

### Description

Compares two client-side image signatures using the weights provided in `attrWeights`.

Unlike the non-static version of this function, the static function can directly operate on signature data without an instance of an `OrdVir` object.

### Parameters

#### **signature1, signature2**

The signatures of the two images.

#### **attrWeights**

A list of weights to apply to each visual attribute. Data type is `VARCHAR2`. The following attributes can be specified, with a value of 0.0 specifying no importance and a value of 1.0 specifying highest importance. You must specify a value greater than zero for at least one of the attributes. The facial attribute is not compatible with any other attributes.

---

Attribute	Description
globalcolor	The weight value (0.0 to 1.0) assigned to the global color visual attribute. Data type is <code>NUMBER</code> . Default is 0.0.

Attribute	Description
localcolor	The weight value (0.0 to 1.0) assigned to the local color visual attribute. Data type is NUMBER. Default is 0.0.
texture	The weight value (0.0 to 1.0) assigned to the texture visual attribute. Data type is NUMBER. Default is 0.0.
structure	The weight value (0.0 to 1.0) assigned to the structure visual attribute. Data type is NUMBER. Default is 0.0.
facial	A value of 1 indicates that this is a facial signature. Data type is NUMBER. Default is 0.0.

**connection**

An object representing a connection to the server.

**Returns**

A floating-point number between 0.0 and 100.0, representing the weighted sum of the distances between the visual attributes.

**Exception**

java.sql.SQLException

**Usage**


---



---

**Note:** When specifying parameter values that include floating-point numbers, you should use double quotation marks (" ") around the value. If you do not, this may result in incorrect values being passed, and you will get incorrect results.

---



---

The connection string of the two image objects must be the same.

```
try{
    byte signature1 = new byte[2000];
```

```
        byte signature2 = new byte[2000];
        .
        .
        .
        float score= OrdVir.score(signature1,signature2,
            "localcolor=1.0",connection);
    }

    catch(SQLException e){
        .
        .
        .
    }
```

---

## similar Method

### Format

```
public int similar( byte [] cmpSignature,
                  string attrWeights,
                  float threshold)
throws SQLException
```

### Scope

public

### Description

Compares the current client-side image to a signature provided in `cmpSignature`, using the weights provided in `attrWeights`.

### Parameters

#### **cmpSignature**

The signature of the image stored in `cmpImg`.

#### **attrWeights**

A list of weights to apply to each visual attribute. Data type is VARCHAR2. The following attributes can be specified, with a value of 0.0 specifying no importance and a value of 1.0 specifying highest importance. You must specify a value greater than zero for at least one of the attributes. The facial attribute is not compatible with any other attributes.

Attribute	Description
globalcolor	The weight value (0.0 to 1.0) assigned to the global color visual attribute. Data type is NUMBER. Default is 0.0.
localcolor	The weight value (0.0 to 1.0) assigned to the local color visual attribute. Data type is NUMBER. Default is 0.0.

Attribute	Description
texture	The weight value (0.0 to 1.0) assigned to the texture visual attribute. Data type is NUMBER. Default is 0.0.
structure	The weight value (0.0 to 1.0) assigned to the structure visual attribute. Data type is NUMBER. Default is 0.0.
facial	A value of 1 indicates that this is a facial signature. Data type is NUMBER. Default is 0.0.

**connection**

An object representing a connection to the server.

**threshold**

The threshold value with which the weighted sum of the distances is to be compared. If the weighted sum is less than or equal to the threshold value, the images are considered to match. The range of this parameter is from 0.0 to 100.0.

**Returns**

The result of the comparison. If the result is less than or equal to the threshold, this method returns 1. Otherwise, this method returns 0.

**Exception**

java.sql.SQLException

**Usage**

```
try{
    OrdVir imageObj = new OrdVir (connection);
    byte signature2 = new byte[2000];
    .
    .
    .
    int result = imageObj.similar(signature2, "localcolor=1.0",connection, 10);
}
```

```
catch(SQLException e){  
    .  
    .  
    .  
}
```

---

## similar Method (static)

### Format

```
public static int similar( byte []    signature1,
                          byte []    signature2,
                          string     attrWeights,
                          float      threshold
                          connection connection)
```

throws SQLException

### Scope

public

### Description

Compares the two client-side image signatures using the weights provided in `attrWeights`.

Unlike the non-static version of this function, the static function can directly operate on signature data without an instance of an `OrdVir` object.

### Parameters

#### **signature1, signature2**

The signatures of the images to be compared.

#### **attrWeights**

A list of weights to apply to each visual attribute. Data type is `VARCHAR2`. The following attributes can be specified, with a value of 0.0 specifying no importance and a value of 1.0 specifying highest importance. You must specify a value greater than zero for at least one of the attributes. The facial attribute is not compatible with any other attributes.

Attribute	Description
globalcolor	The weight value (0.0 to 1.0) assigned to the global color visual attribute. Data type is <code>NUMBER</code> . Default is 0.0.

Attribute	Description
localcolor	The weight value (0.0 to 1.0) assigned to the local color visual attribute. Data type is NUMBER. Default is 0.0.
texture	The weight value (0.0 to 1.0) assigned to the texture visual attribute. Data type is NUMBER. Default is 0.0.
structure	The weight value (0.0 to 1.0) assigned to the structure visual attribute. Data type is NUMBER. Default is 0.0.
facial	A value of 1 indicates that this is a facial signature. Data type is NUMBER. Default is 0.0.

**threshold**

The threshold value with which the weighted sum of the distances is to be compared. If the weighted sum is less than or equal to the threshold value, the images are considered to match. The range of this parameter is from 0.0 to 100.0.

**connection**

An object representing a connection to the server.

**Returns**

The result of the comparison. If the result is less than or equal to the threshold, this method returns 1. Otherwise, this method returns 0.

**Exception**

java.sql.SQLException

**Usage**

```
try{
    OrdVir imageObj = new OrdVir (connection);
    byte signature1 = new byte[2000];
    byte signature2 = new byte[2000];
    .
    .
    .
}
```

```
        int result = imageObj.similar(signature1, signature2,
            "localcolor=1.0",connection, 10);
    }
    catch(SQLException e){
        .
        .
        .
    }
```

## getSignature Method

### Format

```
public byte [] getSignature( )  
throws SQLException
```

### Scope

public

### Description

Reads the signature of the OrdVir object stored on the client side.

### Parameter

None.

### Returns

Returns the signature of the OrdVir object stored on the client side.

### Exception

java.sql.SQLException

### Usage

```
try{  
    OrdVir  imageObj = new OrdVir (connection);  
    .  
    .  
    .  
    byte[] imgSignature = imageObj.getSignature();  
}  
catch(SQLException e){  
    .  
    .  
    .  
}
```

---

## setSignature Method

### Format

```
public void setSignature(byte [] value)
throws SQLException
```

### Scope

public

### Description

Sets the signature field of the client OrdVir object. The signature value must be generated by either the analyze() method or Viisage software.

### Parameter

None.

### Returns

None.

### Exception

java.sql.SQLException

### Usage

```
try{
    OrdVir imageObj = new OrdVir (connection);
    byte[] imgSignature = new byte[2000];
    .
    .
    .
    imageObj.setSignature(imgSignature);
}
catch(SQLException e){
    .
    .
    .
}
```

### 2.2.3 OrdVir Methods for Communication Between the Client and Server

This section presents reference information on the OrdVir methods associated with the communication between the client and server.

The methods described in this section show examples based on the instantiation of an OrdVir object. For the examples in this section, please consult the following Java code:

```
import oracle.ord.media.mediaClass;

public class clientProgram {
    public static void main(String[] args){

        The code in the examples should be placed here.

    }
}
```

In each example, the parameter connection is a connection to an Oracle8i database.

---

## flush() Method

### Format

```
public void flush()  
throws SQLException
```

### Scope

public

### Description

Sends information from the client-side OrdVir object to the server-side OrdVir object.

### Parameter

None.

### Returns

None.

### Exception

java.sql.SQLException

### Usage

```
try{  
    OrdVir imageObj = new OrdVir(connection);  
    .  
    .  
    .  
    imageObj.flush();  
}  
catch(SQLException e){  
    .  
    .  
    .  
}
```

## refresh() Method

### Format

```
public void refresh(boolean forUpdate)
throws SQLException
```

### Scope

public

### Description

Sends information from the server-side OrdVir object to the client-side OrdVir object with or without locking the database row.

### Parameter

#### **forUpdate**

Indicates whether or not the option is selected to be updated. It is set to *true* if it is selected to be updated; *false* otherwise.

### Returns

None.

### Exception

java.sql.SQLException

### Usage

```
try{
    OrdVir imageObj = new OrdVir(connection);
    .
    .
    .
    imageObj.refresh(false);
}
```

```
catch(SQLException e){  
    .  
    .  
    .  
}
```

where:

- *false* indicates the option was not refreshed for the update.

---

# Glossary

## **ADT**

*See* object type.

## **aspect ratio**

Proportions of height versus width.

## **BFILE**

A large object whose value is composed of binary data, and is stored outside of the database in an operating system file. The file itself is not stored in the database, but a pointer to the file is stored in the database. Because they are outside of the database, BFILES are read-only.

## **binary large object**

*See* BLOB.

## **biometrics**

The science of using personal characteristics to verify identity. This method uses a non-intrusive test, such as face or fingerprint recognition, rather than an intrusive test such as DNA or blood type analysis.

## **BLOB**

Binary large object. Large objects are stored in the database tablespace in a way that optimizes space and provides efficient access. Only a pointer to the object is actually stored in the row.

## **content format**

A description of the image data, such as the pixel or color format.

**cropping**

Selecting the portion of an image within a specified rectangle and removing everything outside of that rectangle. *See also* cutting.

**cutting**

Selecting the portion of an image within a specified rectangle to create a subimage. If the subimage is copied to a new image, the effect is a cut. If the subimage replaces the original, the effect is a crop. Use the `Process()` or `ProcessCopy()` procedures to cut or crop an image.

**data service**

The mechanism by which clients, application-specific servers, and database servers can be easily and reliably extended.

**default constructor**

The type of constructor in an Oracle Call Interface (OCI) call that creates an empty instance of the constructor.

**distance**

A measure of how similar two images are. When two images are compared, a distance value for visual attribute and an overall distance value (weighted sum of the attribute distances) are calculated. The distance for each visual attribute can range from 0 (no difference) to 100 (maximum possible difference). Thus, the more similar two images are with respect to a visual attribute, the smaller the distance will be between their scores for that attribute.

**file format**

The file format of the image data, such as BMPF or GIFF.

**foreign images**

An image whose type is not natively supported by Visual Information Retrieval. Instead, the product provides direct access to the pixel data.

**global color**

The visual attribute that represents the distribution of colors within the entire image. This distribution includes the amounts of each color, but not the locations of color.

**IDL**

Interface Definition Language.

**image**

A graphic picture. The source could be a photograph, drawing, or generated image.

**image processing**

A change to the properties of an image, such as through scaling, rotation, or compression.

**local color**

The visual attribute that represents color distributions *and where they occur* in an image, such as the fact that an RGB vector for sky blue occurs in the upper half of an image.

**lossless compression**

A means for reducing the storage space required for an image. The decompressed image is bit-for-bit identical to the original.

**lossy compression**

A means for reducing the storage space required for an image. The decompressed image has less resolution than the original, although this might not be noticeable to the naked eye.

**MIME**

Multi-purpose Internet Mail Extensions. MIME is an Internet specification describing file types. Servers and browsers read the MIME type in the file header and decide what to do with the file, such as displaying it with a viewer or playing it as an audio file.

**Network Computing Architecture**

A fully integrated design integrating text, spatial, and image data. The architecture supports the design, development, installation, and integration of manageable components across entire organizations.

**object relational database**

A database having both object-oriented and relational characteristics. Objects can be defined and stored, and then retrieved using standard relational methods.

**object type**

A data type that encapsulates attributes of the data and methods (functions and procedures) for operating on the data.

An object data type is sometimes referred to as an abstract data type (ADT).

**OCI**

Oracle Call Interface.

**raw pixel format**

An uncompressed image format with a simple, fixed-size header.

**scaling**

A change to the proportions of an image in one or both dimensions. To enlarge an image, scale by a factor greater than one. To shrink an image, scale by a factor between zero and one. Use the Process() or ProcessCopy() procedures to scale an image.

**structure**

The visual attribute that represents the shapes that appear in the image, as determined by shape-characterization techniques, rather than by local shape-segmentation methods (which are affected by problems due to lighting effects and occlusion).

**texture**

The visual attribute that represents the low-level patterns and textures within the image, such as graininess or smoothness. Unlike structure, texture is very sensitive to high-frequency features in the image.

**threshold**

The numeric value used in a comparison to determine whether or not two images match. If the weighted sum of the distances for the visual attributes is less than or equal to the threshold, the images match; if the weighted sum is greater than the threshold, the images do not match.

**weight**

A numeric value assigned to a visual attribute to determine how important that attribute is in determining whether or not two images being compared match. Each weight value can range from 0.0 (no importance) to 1.0 (highest importance), and reflects how sensitive the matching process should be to the degree of similarity or dissimilarity between two images with respect to that attribute.

---

---

# Index

## A

---

aspect ratio, Gloss-1

## B

---

binary large object, Gloss-1  
biometrics, Gloss-1

## C

---

content format, Gloss-1  
content-based retrieval  
    definition, 1-2  
convert()  
    static, 2-12  
cropping images, Gloss-2  
cutting images, Gloss-2

## D

---

data service, Gloss-2

## F

---

file format, Gloss-2  
flush(), 2-28  
foreign images, Gloss-2

## G

---

getSignature(), 2-25

## I

---

IDL, Gloss-2  
image, Gloss-3  
image processing, Gloss-3  
inherited methods, 2-5

## L

---

lossless compression, Gloss-3  
lossy compression, Gloss-3

## M

---

methods  
    convert()  
        static, 2-12  
    flush(), 2-28  
    getSignature(), 2-25  
    inherited, 2-5  
    protected, 2-1  
    public, 2-1  
    refresh(), 2-29  
    score(), 2-14  
        static, 2-16  
    setSignature(), 2-26  
    similar(), 2-19  
        static, 2-22  
MIME, Gloss-3

## N

---

Network Computing Architecture, Gloss-3

## O

---

- object relational database, Gloss-3
- object type, 2-1
- OCI, Gloss-4
- ORDImage, 2-1
- ORDSource, 2-1
- ORDVir
  - methods, 2-4
    - client/server communication, 2-5, 2-27
    - copy operations, 2-6
    - generating SQL queries, 2-7
    - image attributes, 2-5
    - locking, 2-7
    - media type, 2-7
    - processing commands to the external source, 2-6
    - properties attribute, 2-6
    - source attribute, 2-6
    - SQL type, 2-7
    - visual information retrieval attributes, 2-5
  - object type, 2-1

## P

---

- protected methods, 2-1
- public methods, 2-1

## R

---

- raw pixel data format, Gloss-4
- refresh(), 2-29
- retrieval, content-based
  - definition, 1-2

## S

---

- scaling, Gloss-4
- score(), 2-14
  - static, 2-16
- setSignature(), 2-26
- similar(), 2-19
  - static, 2-22
- static methods, 2-12, 2-16, 2-22