



Services API Overview

Contents

Digimarc Services API Overview.....	3
Digimarc Support.....	3
How are Digimarc digital watermarks used?.....	3
Organize Digimarc Digital Watermarks.....	3
Groups/Partitions.....	4
Projects.....	4
Tags.....	4
Services API Credentials.....	5
Enterprise Account.....	5
WalkUp Account.....	5
Services API Environments.....	5
Authentication.....	6
Getting Started.....	7
Additional Features.....	7
Document Conventions.....	7
Date Formats.....	8
Rate Limits.....	8
Resource Usage Limits.....	8
Method Invocation.....	9
Request and Response Body Formats.....	9
Format as a Query Parameter.....	9
Common Response Codes.....	9
Extended Error Information.....	10
Enhance Digital Media.....	16
Step 1: Upload the File.....	16
URL.....	16
Responses.....	16
Step 2: Create the Embed Job.....	17
Headers.....	17
URL.....	17
Responses.....	17
Step 3: Start the Embed Job.....	18
Headers.....	18
URL.....	18
Responses.....	19
Step 4: Check the Embed Job Status.....	20
Headers.....	21
URL.....	21
Response.....	21
Glossary.....	23

Digimarc Services API Overview

This document describes the Services API (version 2.0 and above) exposed by the Digimarc Print & Audio and Packaging modules. The Services API is intended for programmatic access to most of the core features of the Print & Audio and Packaging modules. Applications can use this API to automate the tasks associated with creating a link between media containing a Digimarc digital watermark and the related content.

The Print & Audio and Packaging modules within My.Digimarc (<https://my.digimarc.com>) are the web portals through which customers create and manage Digimarc digital watermarks.

My.Digimarc is a gateway to the Digimarc Platform for partners, customers, and their suppliers that simplifies managing enhancement for packaging, print, audio, and other media. My.Digimarc also provides access to training content, developer tools, and support. My.Digimarc empowers teams of all sizes to reliably and efficiently work with the Platform.

Digimarc digital watermark is a proprietary method for imperceptibly carrying data applied to print, packaging, audio, labels, and everyday objects. It's a two-dimensional data carrier with contrasting elements that vary in appearance. Unlike traditional black-and-white optical codes, digital watermarks are incorporated into printed graphics, making them almost invisible. Digimarc firmware allows mobile devices, inspection cameras, and barcode scanning systems to detect and decode the data.

Note

Historically, the digital watermark has been termed a “service” in the Digimarc Print & Audio and Packaging modules. The Services API and this document retain this term; however, its definition has been refined to mean the digital data structure that represents information about a Digimarc digital watermark.

Digimarc Support

For help using the Services API, contact support.

- [Digimarc Support Website](#)
- [Send email to Digimarc Support](#)

How are Digimarc digital watermarks used?

Digimarc seeks to transform machine-readable code applications in devices such as retail barcode scanners and mobile phones.

Digimarc digital watermark is used in product packaging to provide a more reliable UPC/EAN barcode, and in commercial print and audio applications for content monitoring and more efficient and accurate automatic content recognition. A Digimarc digital watermark carries data in commercial print materials, product packaging, audio files, retail scale labels, and other media. Instead of traditional barcodes, Digimarc technology uses unique, imperceptible identifiers, including signal-rich art on packaging and inaudible signatures in audio files. It can create applications for retail, consumer goods, publishing, movies, music, and more.

Digimarc software works with various scanning devices that use an image-based (optical) scanner. Digimarc has partnerships with major scanner original equipment manufacturers (OEMs), including Datalogic, Honeywell, NCR, and Zebra Technologies, which have integrated Digimarc technology into some of their scanners. Many other scanners can be easily updated with a software upgrade to support digital watermarks.

Digimarc can be enabled on red or white illumination, common among point-of-sale (POS) scanners. Like other two-dimensional (2D) codes, digital watermarks don't work with a laser-only scanner. Some retail scanners feature a hybrid laser and imaging solution that can leverage Digimarc technology.

Organize Digimarc Digital Watermarks

Digimarc offers a few ways to organize your services.

Groups/Partitions

For Print & Audio module users

In addition to projects to organize services, groups (called partitions in the Services API) provide a way to manage collections of users. Groups in the Print & Audio module enable you to limit service access to users of the same group. You can find more information about groups in the *Guide to the Print and Audio Module*.

For Packaging module users

Groups (called partitions in the Services API) provide a way to manage collections of users. Groups in the Packaging module enable you to limit service access to users of the same group. You can find more information about groups in the *Guide to the Packaging Module*.

Projects

For [WalkUp](#) accounts with multiple services, it might be helpful to organize them into projects.

Projects provide a way to define sets of services. A project can represent a magazine edition, product launch, advertising campaign, or playlist. You choose a naming convention that best suits your workflow. Each project contains one or more services.

[Enterprise](#) accounts have a single project per group/partition. Although you can perform some operations on a project, the create, update, and delete Project methods don't apply to Enterprise accounts.

Tags

Tags enable you to organize your packages and are available to all users across all groups.

[Enterprise](#) account members can use tags to organize services further. How you use the tags is entirely up to you. A tag can represent a magazine edition, product launch, advertising campaign, playlist, and so on. You choose a naming convention that best suits your workflow. You can create up to 1000 tags per account. Find more information about tags in the *Guide to the Packaging Module*.

[WalkUp](#) accounts don't support tags. Instead, you can use projects to organize services.

Services API Credentials

Your application requires credentials to submit requests to the Services API.

At [My.Digimarc](#), there are two types of accounts: Enterprise and WalkUp. You can find out your account type programmatically by calling the `accountRead` method and reviewing the `UserExperience` field in the response body. How you get credentials depends on your account type.

Enterprise Account

Enterprise accounts are primarily used with the Packaging module, though they can also use the Print & Audio module. To get credentials for the Print & Audio and Packaging modules, you must have a user account linked to a company account. Contact your Digimarc company account manager to create a Print & Audio module or Packaging module user account. If you don't have a company account, contact Digimarc Support.

WalkUp Account

WalkUp accounts are solely for the Print & Audio module. To get credentials for the Print & Audio module:

- 1 If you don't have an account on My.Digimarc, [contact Digimarc](#) to create one.
- 2 Log in to the Print & Audio module.
- 3 Click the **Developer** tab at the top of the Print & Audio module.
- 4 Scroll down to the *Services API* section.
- 5 Click **Request API Access**. The *Request Services API Access* dialog opens.
- 6 Click **Request**. The *Create a Support Ticket* page opens.
- 7 Fill in the requested information and submit it. A member of Digimarc's customer services team will contact you.

The credentials comprise a unique application name (a "WebAPIUser") and API key combination. You'll receive an email containing the Services API base URL and your API key to use with the application name you provided. The WebAPIUser credentials are set at the Account or Group level. The use of your credentials is discussed in [Authentication](#).

Services API Environments

As stated in the [Digimarc Services API Overview](#), the Services API provides programmatic access to most of the features of the Digimarc Print & Audio and Packaging modules. There are two versions of the modules:

- The **Labs** Print & Audio and Packaging modules enable you to safely develop and test your web services application.
- The **Live** Print & Audio and Packaging modules are the real-world environments in which the production version of your web services application operates.

These two versions of the Print & Audio and Packaging modules run on different servers and require distinct credentials. The table below lists the URLs for the Print & Audio and Packaging modules and their Services API for both Live and Labs versions.

Digimarc Base URLs

Environment	Print & Audio module URL	Packaging module URL	Services API base URL
Labs	labs-portal.digimarc.net	labs-central.digimarc.com	labs-api.digimarc.net
Live	portal.digimarc.net	central.digimarc.com	api.digimarc.net

You do all your development and testing on the Labs Services API. When your app is ready for production, you can start using the Live Services API. The information in this document applies equally to both Live and Labs versions, except where noted.

Authentication

Authentication is required to use the Services API, and authentication requires the credentials obtained through the Digimarc Print & Audio and Packaging modules.

You submit your credentials using HTTP Basic Authentication (encrypted over HTTPS). The format of the header for Basic Authentication is:

```
Authorization: Basic <string>
```

where <string> is the base-64-encoding of <Application Name>:<API Key>.

For example, if your Application Name is myWebApp and your API Key is 6M8+pjLZhgT7Jtpri0ZeAh+d99hG1nMP, combining these gives:

```
myWebApp:6M8+pjLZhgT7Jtpri0ZeAh+d99hG1nMP
```

and the base-64 encoding of this combination is:


```
bXlXZWJBcHA6Nk04K3BqTFpoZlQ3SnRwcmkwWmVBAcTkOTloRzFuTVA=
```

Your authentication header would then be:

```
Authorization: Basic bXlXZWJBcHA6Nk04K3BqTFpoZlQ3SnRwcmkwWmVBAcTkOTloRzFuTVA=
```

You must provide your credentials every time you make an API request. No sessions are used, and there is no option for unsecured communication.

For efficient use of the Services API, send your user credentials preemptively. Some HTTP clients omit the authentication header and make a second request with the authorization header when they get a 401 Unauthorized response that includes a WWW-Authenticate header. Send your credentials directly with each request to avoid unnecessary round trips.

To experiment with the Services API methods using a third-party tool like  [Postman](#), use your [Labs Print & Audio or Packaging module](#) username and password. This avoids creating unnecessary and potentially costly projects and services in your Live account.

After your application is thoroughly tested and ready for production, you can get production credentials from the Live Digimarc Print & Audio module by following the steps in [Services API Credentials](#).

Getting Started

To use and understand the Services API, you must be familiar with the HTTP protocol and RESTful web services. The Services API allows actions on resources by utilizing standard HTTP verbs. Your development platform/tooling must support access to HTTP headers and verbs, Basic authentication, SSL, and XML or JSON parsing.

The table below describes the operations that are accessible through the Services API.

Watermark Support	Create	Read	Update	Enhance	Delete
Print	Yes	Yes	Yes	Yes	No
Packaging with GTIN	Yes	Yes	Yes	No	No
Packaging with GTIN+	No	Yes	Yes	No	No
Audio	Yes	Yes	Yes	Yes	No

Additional Features

- Search services in your account
- Organize your services with [projects or tags](#)
- View account and user information

See the *Guide to the Print and Audio Module* or the *Guide to the Packaging Module* for more information about these features.

Document Conventions

Following are the standards and conventions used in this documentation. See the Glossary for terms related to digital watermarking.

Convention	Description
Numeric format	The fractional part of a numeric value is represented by the period character, such as pi = [~] 3.14
Elements in the user interface	Formatting: Text in bold
Titles of panels and dialogs	Formatting: Text in italics
User input (text that a user types)	Formatting: Text in a monospace font
File and directory paths	Formatting: Text in a monospace font
Variables or variable data	Formatting: Text in italics, often within angle brackets (<>)

Tip

Tips provide helpful information.

Note

Notes bring additional information to your attention.

Warning

Warnings point out important information not to be overlooked.

Caution

Cautions point out critical information that could be detrimental or essential to success.

Date Formats

All dates used in request and response bodies must be in ISO 8601 extended format. Where time is specified, it's always UTC time.

For example:

```
2011-04-12T13:00:00Z
```

Where noted, some fields ignore the time if it's included. Also, where noted, some fields allow for a time offset to be specified, for example:

```
2011-04-12T13:00:00-07:00
```

When time offset is positive, for example: 2014-04-12T13:00:00+04:00, the plus sign (+) must be replaced by its URL-encoded value (%2B): 2014-04-12T13:00:00%2B04:00.

Rate Limits

The Services API enforces usage limits to avoid individual applications degrading the overall user experience.

The API tracks usage for short and long intervals and has different limits for each associated with every set of credentials. The time intervals used for short and long usage tracking are currently defined as per minute and per hour; however, these intervals could change over time as overall usage patterns change. Intervals start with the first incoming request and are not tied to a clock-based minute or hour boundary.

Every call to the Services API returns the current rate-limiting status in custom HTTP headers of the response. The headers are named `X-RateLimit-Short` and `X-RateLimit-Long`. Both headers use the same format, as shown in the following examples:

```
X-RateLimit-Short: Limit=300; Remain=245; Expires=45
```

```
X-RateLimit-Long: Limit=7500; Remain=7401; Expires=3219
```

where:

- **Limit** indicates the total number of requests allowed in the interval.
- **Remain** indicates the number of requests remaining in the interval.
- **Expires** indicates the number of seconds before the interval expires.

In the example of the short interval above, 300 requests are allowed per interval, with 245 remaining in the next 45 seconds. After an interval expires, the Remain value is reset to the limit.

If you exceed the limit in an interval, you receive HTTP 429 Too Many Requests errors with Rate Limit Exceeded in the response body until the next rate-limited period begins. In this case, the Remain portion of one (or both) of the headers is 0, and Expires indicates how many seconds you must wait before the next request will succeed.

Additionally, each account on the Labs server has a limit on the total number of services that can be active at a time. When this limit is reached, a 403 Forbidden error is returned with Service Limit Exceeded in the response body. By default, this limit is set based on your expected usage. This service limit doesn't apply to the Live server environment.

Resource Usage Limits

Resource usage limits are also enforced on methods identified as overly resource-intensive. These include the Service Activity methods and the Enumerate Services method. It's possible to exceed these limits before reaching the short or long rate limit. In cases where the resource usage limit is exceeded, further calls to the monitored methods result in a 503 Service Unavailable response with Method usage suspended due to resource constraints in the response body. Once resource usage has stabilized for the calling user, additional calls are permitted.

These limits are intended to protect the system from poorly coded applications and retain the system's overall usability, not to unduly restrict you as a user. If these limits are overly restricting, contact your Digimarc account representative to discuss higher limits.

Method Invocation

To call a Services API method, combine the base URL with the URL snippet for the desired method to produce the request URL. For example:

Base URL:	labs-api.digimarc.net
Method URL snippet:	v2/service/12345
Request URL:	https://labs-api.digimarc.net/v2/service/12345

The v2 URL component must be included in every request.

String parameters passed to the API are trimmed of any leading or trailing white space characters. The API prohibits the use of the greater-than (>) and less-than (<) characters in string parameters. When these or other special characters are included in a string parameter passed to a web service method, the method returns status code 400 Bad Request.

Data returned by the API isn't HTML-encoded. For proper display in an HTML application, HTML-encode the data before displaying it.

Request and Response Body Formats

All methods of the Services API support XML and JSON formats in the Request and Response bodies. The caller can control the format using standard HTTP content negotiation through the Content-Type and Accept headers.

For example, consider the following HTTP headers in a request:

```
Content-Type: application/xml
```

```
Accept: application/xml
```

This request results in the method interpreting the request body as XML and instructs it to return any response in XML format. JSON format is similarly supported using application/json.

When using XML format for requests, the order of request parameters **must** match the order specified in that method's request parameters description. Similarly, the response order will match the documented order.

Data returned by the API isn't HTML encoded. For use in an HTML application, we recommend the data be HTML encoded before being displayed.

Format as a Query Parameter

All methods allow the caller to specify the response format on the query string. These examples show the use of the format query parameter to request JSON or XML responses:

```
https://example.digimarc.net/v2/object?format=json
```

```
https://example.digimarc.net/v2/object?name=value&format=xml
```

This query parameter is provided as a convenience for testing method calls within a browser. Don't rely on it when calling the API programmatically because it's not always possible to override the standard HTTP content negotiation.

Common Response Codes

Standard HTTP status codes and messages are returned by API methods for success and failure conditions. For failure conditions, the message body includes a short description of the first error encountered.

The following HTTP status codes can be returned by any of the API methods (in addition to error codes described in each method's reference).

Status Code	Status Message	Description
400	Bad Request	Invalid parameter
401	Unauthorized	Caller isn't authorized
403	Forbidden	Operation not allowed

Status Code	Status Message	Description
409	Conflict	There was a conflict in the current state of the resource.
429	Too Many Requests	Rate limit exceeded. See Rate Limits .
500	Internal Server Error	Unexpected internal error. Please report this error to Digimarc.
503	Service Unavailable	The web service is temporarily unavailable (under maintenance) OR Method usage was suspended due to resource constraints.

Extended Error Information

In addition to the standard errors, the Services API can display extended error information. This information isn't displayed by default. To view extended errors when they occur, include the **exerror** query parameter set to 1 (true).

For example:

```
curl -X GET "https://labs-api.digimarc.net/v2/account/1234?exerror=1"
```

When **exerror** is enabled, failure conditions return a `WebApiErrorResponse` response with these members:

```
{
  HttpStatus integer (int32)
  The HTTP status code returned by the server

  Code string
  The error code returned by the server

  CodeDescription string
  The description of the first error that occurred

  Occurred string
  The date and time the error occurred (UTC)

  Source string
  The part of the system that triggered the error, such as a parameter name or entity identifier
}
```

See the error categories that follow for details.

Account Errors

ACT_AccountExists	An attempt was made to create or update an account with an existing company name
ACT_AccountNotFound	The requested account was not found
ACT_AccountPartitionNotFound	The requested partition was not found
ACT_AccountPartitioningNotEnabled	An attempt was made to call the partitioning method on an account that isn't partition-enabled
ACT_AccountNotActive	An attempt was made to update the account data for a disabled account
ACT_UpdateAccountBeforeDisabling	An attempt was made to update an account while disabling it
ACT_PartitionLimitExceeded	The partition limit for the account has been exceeded
ACT_ContactCreateRequestRequired	An attempt was made to create an account without contact information
ACT_DefaultBannerImageNotFound	An attempt was made to create a partition with an existing name
ACT_DuplicatePartitionName	An attempt was made to create a partition with an existing name

ACT_ProvinceNotFound	The province specified for the account contact was not found
ACT_CountryNotFound	The country specified for the account contact was not found
ACT_CannotDeletePartitionWithServiceInEnterpriseAccount	An attempt was made to delete a partition containing a service for the Enterprise account
ACT_CannotDeleteLastPartitionInEnterpriseAccount	An attempt was made to delete the last partition for the Enterprise account
ACT_CannotChangePermissionsForEnterpriseAccount	An attempt was made to change a user's audio/visual or packaging permissions for the Enterprise account
ACT_ServiceTypeNotSupportedForEnterpriseAccount	An attempt was made to create a service of an unsupported type for the Enterprise account
ACT_ServiceTypeNotSupportedForWalkUpAccount	An attempt was made to create a service of an unsupported type for the WalkUp account
ACT_WalkUpAccountsDoNotSupportPackagingPermissions	An attempt was made to set unsupported packaging permissions for the WalkUp account

Availability Errors

GEN_CacheConnectivityIssues	The cache service failed the availability check
GEN_DatabaseConnectivityIssues	The database is experiencing connectivity issues
GEN_DependencyConnectivityIssues	An unexpected connectivity error occurred in a dependent component
GEN_RateLimitLimitExceeded	The reporting service isn't available
GEN_UsageLimitExceeded	The service or method isn't available due to resource constraints
RPT_ReportingAvailability	The service isn't available due to rate limit constraints

Embed Job Errors

EMB_EmbedJobNotFound	The requested enhancement (embed) job was not found
EMB_UnmarkedFileNotFound	An unenhanced (original) file was not found for the specified enhancement (embed) job
EMB_MarkedFileNotFound	An unenhanced (original) file was not found for the specified enhancement (embed) job
EMB_EmbedReportNotFoundForService	The enhancement report was not found for the specified service
EMB_InvalidServiceType	The specified service type does not support enhancement
EMB_FileTypeNotSupported	The specified file type does not support enhancement
EMB_EmbedJobInvalidOperation	The enhancement (embed) job is in an invalid state for the requested operation
EMB_NotReadyForEmbed	An attempt was made to start an enhancement (embed) job that isn't ready for enhancement
EMB_AudioEmbedOptionsRequired	Audio enhancement options were not specified
EMB_ImageEmbedOptionsRequired	Image enhancement options were not specified
EMB_JobCompletionConflict	The current job status prevents marking the job as completed
EMB_JobResetConflict	The current job status prevents a reset
EMB_JobUpdateConflict	The current job status prevents an update
EMB_ImageMetadataAlphaChannel	Enhancement of alpha channel images isn't supported

Embed Report Errors

EMB_ParameterNotValidForServiceType	The enhancement report parameter isn't valid for the service type
-------------------------------------	---

File Errors

FIL_FileExists	An attempt was made to commit or save a file with an existing file ID
FIL_FileNotFound	The requested file was not found
FIL_FileStreamRequired	A file stream is required for the requested operation
FIL_FileAlreadyCommitted	An attempt was made to commit a file that has already been committed
FIL_FileLimitExceeded	The file limit for the account has been exceeded
FIL_ContentLengthMismatch	The specified content length does not match the stream length
FIL_ContentLengthExceedsMax	The specified MD5 checksum does not match the expected value
FIL_ContentMd5Mismatch	The file content length exceeds the maximum size allowed

General Errors

GEN_RequestBodyRequired	The request body was not supplied
GEN_RequestContentInvalid	The request body was not well formed
GEN_NoUpdateParametersSpecified	A request body parameter required for the update is empty
GEN_UserNotAuthorized	The user isn't authorized to perform the requested operation
GEN_OperationNotAllowed	The requested operation isn't allowed (for example, trying to extend a non-extendable service)
GEN_MethodNotAllowed	The requested method isn't supported
GEN_EndpointNotFound	The requested method does not exist
GEN_ImageMetadataNotSupported	The specified file isn't a recognized image file
GEN_ImageMetadataCorrupt	The image metadata could not be parsed
GEN_InternalPropertiesAccessForbidden	The user isn't authorized to access internal properties

Media Service Errors

SVC_ServiceLimitExceeded	The service limit for the account has been exceeded
SVC_DemoServiceLimitExceeded	The demo service limit for the account has been exceeded
SVC_ServiceStatusChangeNotAllowed	The service status cannot be changed
SVC_ServiceNotFound	The requested service was not found
SVC_ServiceNotFoundForPayload	A service was not found for the specified payload
SVC_ServiceThumbnailNotFound	The service thumbnail was not found
SVC_ServiceTypeNotVisual	The requested operation is not valid for a non-visual service
SVC_ServiceNotPackaging	The requested operation is not valid for a non-packaging service
SVC_HumanReadableIdInUse	The specified human-readable ID (HRID) is in use by another service
SVC_HumanReadableIdLengthInvalid	The human-readable ID (HRID) length is invalid
SVC_HumanReadableIdCheckDigitInvalid	The human-readable ID (HRID) check digit is invalid
SVC_PayloadLengthInvalid	The payload length is invalid
SVC_PaymentSetupIdRequired	A payment setup ID is required for the requested operation
SVC_ServiceTypeInvalid	The service type is invalid for the requested operation
SVC_SourceProjectEmpty	The source project ID for the service move operation is empty
SVC_SourceProjectIdAndServiceIdListMutuallyExclusive	Either the project ID or the service ID list can be specified for a service move operation but not both
SVC_DestinationAccountSameAsSourceAccount	The source account is the same as the destination account in the requested service move operation

SVC_GtinSearchRequiresPackagingTypeFilter	An attempt was made to filter services by GTIN for a non-packaging service type
SVC_PayloadChangeForbidden	An attempt was made to change the payload on a non-packaging service type
SVC_IsDemoExtendNotAllowed	Demo services cannot be extended
SVC_AutoExtendNotAllowed WhenServiceIsDemo	Demo services cannot be auto extended
SVC_IsDemoNotAllowedForThreeYear SubscriptionType	Demo services do not support three-year subscriptions
SVC_StartDateTooEarly	The start date is earlier than the earliest allowed date
SVC_StartDateLaterThanEndDate	The start date is later than the end date
SVC_EndDateLaterThanExpires	The start date is later than the expiration date
SVC_EndDateEarlierThanStartDate	The end date is earlier than the start date
SVC_BannerImageNotFound	A banner image was not found for the service
SVC_ImageDimensionsInvalid	The image dimensions are invalid for the requested operation
SVC_ImageFileTypeInvalid	The image file type is invalid for the requested operation
SVC_ImageMustBeJpg	Only JPG images are allowed for the requested operation
SVC_TooManyServiceIds	More than 1000 service IDs were provided
SVC_ServiceTypeDoesntSupportBarcodeType	The provided barcode type is invalid for the service type
SVC_PayloadDoesntSupportBarcodeType	The provided barcode type is invalid for the human-readable ID (HRID)
SVC_BarcodeTypeInvalid	An invalid barcode type was provided
SVC_UserExperienceServiceTypeMismatch	The service type is not supported for the account
SVC_CannotChangeSubscriptionType ToOneYear	The service subscription type cannot be changed to a one-year subscription
SVC_CannotChangeSubscriptionType ToThreeYear	The service subscription type cannot be changed to a three-year subscription

Payoff Errors

GEN_DuplicatePayoff	The service can have only one of each payoff type
---------------------	---

Project Errors

PRJ_ProjectNotEmpty	Only empty projects can be deleted
PRJ_ProjectNotFound	The requested project was not found
PRJ_EnterpriseAccountsDoNotSupportProjects	Projects are not supported for Enterprise accounts.

Report Errors

RPT_DateRangeInvalid	The provided date range is invalid
RPT_ProjectIdAndServiceIdAreMutuallyExclusive	The project ID and service ID may not be specified in the same request
RPT_ProjectLimitExceeded	The number of projects specified exceeds the limit

Resolver SDK User Errors

RSU_UserExists	The user with the application name specified in the create request already exists
RSU_UserLimitExceeded	The maximum number of users has been exceeded
RSU_TrialUserLimitExceeded	The maximum number of trial users has been exceeded
RSU_UserNotFound	The requested user was not found

Search Errors

SER_InvalidQuery	The provided query was not valid
SER_InvalidSources	The provided sources were not valid
SER_TooManyTerms	The provided query has too many terms (maximum is five)

Tag Errors

TAG_TagNotFound	The requested tag was not found
TAG_TagNameAlreadyExists	The tag name is already in use
TAG_MaximumNumberOfTagsExceeded	The tag limit for the account has been exceeded
TAG_TagsOnlySupportedForEnterpriseAccounts	Tags are supported only for Enterprise accounts
TAG_MaxNumberOfTagsOnMediaServiceExceeded	The tag limit for the service has been exceeded

User Errors

PTU_UserNotFound	The requested user was not found
PTU_UserExists	A user with the name specified in the create request already exists
PTU_UserIdAndNameAreMutuallyExclusive	Both the user ID and the user name cannot be specified in the same request
PTU_UserIdAndExternalUserIdAreMutuallyExclusive	Both the user ID and the external user ID cannot be specified in the same request
PTU_UserNameAndExternalUserIdAreMutuallyExclusive	Both the user name and the external user ID cannot be specified in the same request
PTU_UsersTheLastUserInAccount	An attempt was made to remove the last user from an account
PTU_UsersTheLastPrivilegedUserInAccount	An attempt was made to delete the last privileged user (AccountManager, Admin, Portal) from an account

Validation Errors

GEN_StringValueRequired	A required string value is null or consists solely of white space
GEN_StringValueInvalid	The string value is not valid (for example, it contains an invalid URL format, or it's not a valid enumerated type)
GEN_StringValueTooShort	The string does not meet the minimum length requirement (typically seen in change password requests)
GEN_StringValueTooLong	The string exceeds the maximum length allowed
GEN_StringValueInvalidCharacters	The string contains invalid characters
GEN_NumericValueRequired	The required numeric value is blank or zero
GEN_NumericValueInvalid	The numeric value cannot be parsed
GEN_NumericValueBelowMin	The numeric value is less than the minimum allowed
GEN_NumericValueAboveMax	The numeric value exceeds the maximum allowed
GEN_BooleanValueRequired	The required boolean value is blank
GEN_BooleanValueInvalid	The boolean value cannot be parsed
GEN_DateTimeValueRequired	The required date/time value is blank or zero
GEN_DateTimeValueInvalid	The date/time value cannot be parsed
GEN_DateTimeValueBelowMin	The date/time value is earlier than allowed
GEN_DateTimeValueAboveMax	The date/time value is later than allowed

Web API User Errors

WAU_UserExists	A user with the application name specified in the create request already exists
WAU_UserNotFound	The requested user was not found
WAU_UserLimitExceeded	The maximum number of users has been exceeded

Enhance Digital Media

This section describes how to enhance a digital media file through the Digimarc Services API. The process is the same for audio and image files.

The HTTP headers for all the methods in the following steps must include the Authorization header. For more details, see [Authentication](#).

The following steps illustrate enhancing a file through the Services API:

[Step 1](#): Upload the media file.

[Step 2](#): Create an embed job.

[Step 3](#): Start the embed job.

[Step 4](#): Check the embed job status.

Step 1: Upload the File

To begin, send a POST request to the `/file/start` endpoint. This sets up the internal structure to receive the image or audio file. For embed jobs, you can upload files under 150MB. These files are available for 30 days.

The response includes a property, `UploadUrl`, which is an [Azure Blob SAS URL](#) that's valid for one hour. Use the `UploadUrl` with [AzCopy](#) to upload the file.

```
azcopy 'myfilename.png' '<UploadUrl>'
```

Finally, you must call the `/file/finish` endpoint, which is provided as the `FinishURL` property in the `/file/start` response.

Warning

If you don't call `/file/finish`, the uploaded file expires and can't be enhanced, downloaded, or included in a workflow job.

URL

```
curl -X POST "https://labs-api.digimarc.net/v2/file/start?account={account}&fileName={fileName}&fileSize={fileSize}"
```

where:

account * *string*

The unique identifier for the account.

fileName * *string*

The name of the file being uploaded. This parameter doesn't need to be unique, but we recommend it be URL-encoded.

fileSize * *string*

The size of the file (in bytes) being uploaded.

Responses

Successful execution returns:

Status Code	Description
200	Success

Response Body

When `/file/start` is successful, the following properties are returned:

```
{
  UploadUrl string
  The Azure Blob SAS URL for uploading the file.

  FinishURL string
  The URL to call after the file upload has finished.

  FileId string
  The unique identifier for the new file. Note this ID for use in Step 2: Create the Embed Job.
}
```

Errors

Potential error responses are:

Status Code	Description
400	Invalid parameter
404	Account not found

Step 2: Create the Embed Job

In this step, you create a job using the `POST /embed/` endpoint.

Headers

Only the [authorization](#) header is required.

URL

```
curl -X POST "https://labs-api.digimarc.net/v2/embed/{fileId}/{serviceId}?account={account}"
```

where:

fileId * *string*

Identifies the original, unenhanced (also called *unmarked*) file. You get this after the file is uploaded (see [Step 1: Upload the File](#)).

serviceId * *string*

Identifies the service with which to enhance the file. You can get the `serviceId` value by enumerating the services in your project and choosing the `Id` value for the service you want to embed.

account * *string*

The unique identifier for the account.

* indicates a required field

Responses

Successful execution returns:

Status Code	Description
201	Success

Response Body

When successful, this method also returns the following object:

```
{
  id integer (int32)
  The unique identifier for the embed job.
}
```

Note the embed job Id for the next steps.

Errors

Potential error responses are:

Status Code	Description
400	Invalid parameter
403	Invalid service type. Must be an audio or visual service.
404	Service, file, or account not found

Step 3: Start the Embed Job

In this step, you start the embed job to enhance the uploaded file with the selected service.

To get the values for the request body parameters, call the optional GET /embed/options endpoint first. If you don't supply the request body, the default values are used (as determined within your account).

Headers

Only the [authorization](#) header is required.

URL

```
curl -X PUT "https://labs-api.digimarc.net/v2/embed/{id}?account={account}"
```

where:

id * *string (int32)*

The unique identifier for the embed job you want to start. This is the Id returned in [Step 2: Create the Embed Job](#).

account * *string*

The unique identifier for the account.

* indicates a required field

Request body

The request body is optional. When the request body is omitted, the default values are used.

For an Image File

```
embedOptions {
```

```

ImageOptions object {
  WatermarkStrength integer (int32)
  The watermark strength to use for the print environment (valid values: the StrengthOptions returned from the embedJobOptionsRead method; default: WatermarkStrength set in the account).

  WatermarkResolution number (double)
  The watermark resolution to use for the print environment. Use the WatermarkResolution based on your print environment.

  PrintResolution number (double)
  The print size resolution (dots per inch) to use. Use the PrintResolution in a valid print size entry that is closest to your desired image print size
}
}

```

For an Audio File

```

embedOptions {
  AudioOptions object {
    WatermarkStrength integer (int32)
    Watermark strength to use for the audio file (valid values: [1-4]; default: the WatermarkStrength returned from the embedJobOptionsRead method).
  }
}

```

Responses

Successful execution returns:

Status Code	Description
200	Success

Response Body

```

{
  JobId integer (int32)
  The unique identifier for the embed job obtained in the previous step.

  ServiceId integer (int64)
  The unique identifier for the service associated with the embed job.

  EmbedStatus string
  The status of the embed job: ReadyForUpload, ReadyForEmbed, EmbedPending, EmbedPreProcessing, EmbedInProgress, EmbedPostProcessing, EmbedComplete.

  EmbedProgress integer (int32)
  The percentage complete when EmbedStatus is EmbedInProgress, zero otherwise.

  UnmarkedFileId string
  The unique identifier for the unenhanced file; null if the unenhanced file doesn't exist.

  UnmarkedFileName string
  The name of the unenhanced file.

  UnmarkedCreated string
  The date and time the unenhanced file was created.
}

```

MarkedFile *string*

The unique identifier for the enhanced file; null if the enhanced file doesn't exist.

MarkedCreated *string*

The date and time the enhanced file was created.

SubmitDate *string*

The date and time the embed job was submitted for processing.

CompletionDate *string*

The date and time the embed job processing was completed.

ResultCode *integer (int32)*

The embed job result code:

- 0 - Success
- 1 - General Error
- 3 - No audio stream
- 4 - Too many media streams
- 5 - Unsupported format
- 6 - Re-sample required
- 7 - Already watermarked
- 8 - Invalid bit depth
- 9 - Invalid duration - minimum not met
- 10 - Invalid duration - exceeds maximum
- 11 - Embed did not result in a readable watermark
- 253 - Embed job failed and will be retried
- 254 - Embed job failed in all attempts
- 255 - Embed job failed and cannot be retried

EmbedStatusToken *string*

A token used to represent the current embed job status.

}

All date fields are in ISO 8601 extended format. Nullable values are null in this response body; they are updated as appropriate in the next step.

Save the EmbedStatusToken for use in the next step.

Errors

Potential error responses are:

Status Code	Description
400	Invalid parameter
404	Embed job or account not found
409	Not ready for embedding

Step 4: Check the Embed Job Status

In this step, you will check the status of the embed job to determine whether the enhancement operation was successful.

For best results, call this method in a loop that continues until the job is complete. Completion is signaled when the returned *embedStatus* is EmbedComplete.

Headers

Only the [authorization](#) header is required.

URL

```
curl -X GET "https://labs-api.digimarc.net/v2/embed/{embedId}?account={account}"
```

where:

embedId * *integer (int32)*

identifies the job used in the previous steps.

account * *integer (int32)*

identifies the account.

* indicates a required field

This method returns when the status of the job changes or the timeout is reached.

Response

Successful execution returns:

Status Code	Description
200	Success

Response Body

The response body has the same structure returned in the previous step with the following fields updated:

EmbedStatus *string*

The status of the embed job. Possible values are: ReadyForUpload, ReadyForEmbed, EmbedPending, EmbedPreProcessing, EmbedInProgress, EmbedPostProcessing, EmbedComplete. For embed jobs whose status is EmbedComplete, the ResultCode indicates the success or failure of the file enhancement.

EmbedProgress *integer (int32)*

The percentage complete when EmbedStatus is EmbedInProgress, zero otherwise.

ResultCode *integer (int32)*

The embed job result code. The ResultCode is 0 when no error has occurred; errors return a non-zero resultCode.

EmbedStatusToken *string*

A token used to represent the current embed job status. When calling the beginEmbedGetStatus function again to update the job status, set the **token** query parameter to the value of this EmbedStatusToken. Until the embedStatus is EmbedComplete or the resultCode is non-zero, the loop should continue with the latest inserted into the request URL. See beginEmbedGetStatus for the errors associated with a non-zero resultCode.

When the file enhancement is finished (the **EmbedStatus** is set to EmbedComplete, and the ResultCode is set to 0), the following fields are updated in the final response body:

markedFileId *string*

identifies the enhanced file.

markedCreatedDate *string*

is the date the enhanced file was created.

completionDate *string*

is the date the enhancement was completed.

Now, the marked file (identified by **markedFileId**) can be downloaded for publication or broadcast.

For more response body fields, see `beginEmbedGetStatus`.

Errors

Potential error responses are:

Status Code	Description
400	Invalid parameter
404	Embed job or account not found

Glossary

artwork

The elements of design and text that makeup packaging, labels, or other visual media, in which Digimarc Barcode is or will be included.

barcode

The generic term for an optically machine-readable pattern that encodes data in a way that is generally not recognizable to human observation. Barcodes include one-dimensional symbols that encode data into a machine-readable pattern of adjacent, varying width, parallel, dark rectangular bars and pale spaces, and 2-Dimensional Symbols that encode data in the X and Y-axes (sometimes referred to as a 2D barcode), for example, in the case of a QR Code, Data Matrix, and Digimarc digital watermarks.

Digimarc Barcode

The Digimarc digital watermark.

Digimarc digital watermark

A novel data carrier that encodes data in media in ways that are generally imperceptible to people, permitting the carrier to be repeated many times over the surface of the Enhanced media. It delivers unprecedented ease of use, reliability, and efficiency in identifying the media due to massive mathematical and graphical redundancy.

digital media

Digital media includes images and audio tracks.

digital watermark

In this guide, digital watermark refers specifically to the Digimarc digital watermark.

embed

The process to integrate Digimarc digital watermark into a label or package artwork, enabling it for scanning. See enhance.

enhance

The process to integrate Digimarc digital watermark into a label or package artwork, enabling it for scanning.

enhancement

The result of creating and integrating a Digimarc digital watermark into product packaging and other substrates.

group

A set of projects that can be accessed only by members of the group.

partition

Partitions provide a way to manage collections of users and limit service access to users of the same partition. Partitions are called Groups in the Print & Audio and Packaging modules.

partitions

Partitions provide a way to manage collections of users and limit service access to users of the same partition. Partitions are called Groups in the Print & Audio and Packaging modules.

payload

The digital data encoded in a data-carrying signal (such as a Digimarc Barcode or UPC barcode). The payload is generally represented by a string ID.

payoff

A payoff is typically a URL representing a web experience for an end user who has detected the associated payload in digital media using a mobile application that incorporates the Digimarc Mobile SDK (DM SDK). A payoff is represented as a property of a service.

project

A project contains a group of related services (Digimarc digital watermarks). Projects are applicable only to online (Services API) applications and are supported only for WalkUp accounts. For example, a project might contain all the services for a single issue of a magazine. If 50 images in the issue are to be connected to 50 different payoffs, then 50 services are defined in the project. A unique payload is assigned to each service to be embedded in the corresponding image. Every project is automatically assigned a unique identifier. In the DCE SDK, a project is represented as a property of a service.

resolve

Resolve is the process by which the Digimarc Resolver service performs a lookup on metadata carried in a Digimarc digital watermark and redirects the user to a URL or image.

service

An online representation of metadata corresponding to a data carrier in the Digimarc Platform. Services are the central data structure in the Digimarc Platform, defined as extensible data containers that are bound to the encoded data value (payload). They have basic fields including name, description, and tags. More dynamic data, such as a URL response, can be stored and accessed by the resolver. The service is associated with the payload and CPM Path, which are then encoded into a data carrier. The data packet response from the resolver to a device is called the payoff.

services

The online representation of metadata corresponding to a data carrier in the Digimarc Platform. Services are the central data structure in the Digimarc Platform, defined as extensible data containers that are bound to the encoded data value (payload). They have basic fields including name, description, and tags. More dynamic data, such as a URL response, can be stored and accessed by the resolver. The service is associated with the payload and CPM Path, which are then encoded into a data carrier. The data packet response from the resolver to a device is called the payoff.

tags

Tags enable you to organize your packages and are available to all users across all groups.