Interface: DockerDesktopClient

- Properties
  - backend
    - Inherited from
  - extension
    - Inherited from
  - desktopUI
    - Inherited from
  - host
    - Inherited from
  - docker
    - Inherited from
- Container Methods
  - listContainers
    - Parameters
    - Returns
    - Inherited from
- Image Methods
  - listImages
    - Parameters
    - Returns
    - Inherited from
- Navigation Methods
  - navigateToContainers
    - Returns
    - Inherited from
  - navigateToContainer

- Parameters
- Returns
- Inherited from
- navigateToContainerLogs
  - Parameters
  - Returns
  - Inherited from
- navigateToContainerInspect
  - Parameters
  - Returns
  - Inherited from
- navigateToContainerStats
  - Parameters
  - Returns
  - Inherited from
- navigateToImages
  - Returns
  - Inherited from
- navigateToImage
  - Parameters
  - Returns
  - Inherited from
- navigateToVolumes
  - Returns
  - Inherited from
- navigateToVolume
  - Parameters
  - Returns
  - Inherited from

- Other Methods
  - execHostCmd
    - Parameters
    - Returns
    - Inherited from

- spawnHostCmd
  - Parameters
  - Returns
  - Inherited from
- execDockerCmd
  - Parameters
  - Returns
  - Inherited from
- spawnDockerCmd
  - Parameters
  - Returns
  - Inherited from
- openExternal
  - Parameters
  - Returns
  - Inherited from

- Toast Methods
  - toastSuccess
    - Parameters
    - Returns
    - Inherited from
  - toastWarning
    - Parameters
    - Returns
    - Inherited from
  - toastError
    - Parameters
    - Returns
    - Inherited from

An amalgam of the v0 and v1 interfaces of the Docker Desktop API client, provided for backwards compatibility reasons. Unless you're working with a legacy extension, use the v1 type instead.

Properties

backend

• `Readonly` **backend**: `undefined` | `BackendV0`

The `window.ddClient.backend` object can be used to communicate with the backend defined in the vm section of the extension metadata. The client is already connected to the backend.

> [!WARNING]
>
> It will be removed in a future version. Use extension instead.

Inherited from

DockerDesktopClientV0.backend

---

extension

• `Readonly` **extension**: `Extension`

The `ddClient.extension` object can be used to communicate with the backend defined in the vm section of the extension metadata. The client is already connected to the backend.

Inherited from

DockerDesktopClientV1.extension

---

desktopUI

• `Readonly` **desktopUI**: `DesktopUI`

Inherited from

DockerDesktopClientV1.desktopUI

---

host

• `Readonly` **host**: `Host`

Inherited from

DockerDesktopClientV1.host

---

docker

• `Readonly` **docker**: `Docker`

Inherited from

DockerDesktopClientV1.docker

Container Methods

listContainers

▸ **listContainers**(`options`): `Promise`<`unknown`>

Get the list of running containers (same as `docker ps`).

By default, this will not list stopped containers. You can use the option `{"all": true}` to list all the running and stopped containers.

```
const containers = await window.ddClient.listContainers();
```

> [!WARNING]
>
> It will be removed in a future version. Use listContainers instead.

Parameters

Name	Type	Description

`options`	`never`	(Optional). A JSON like `{ "all": true, "limit": 10, "size": true, "filters": JSON.stringify({ status: ["exited"] }), }` For more information about the different properties see the Docker API endpoint documentation.

Returns

`Promise`<`unknown`>

Inherited from

DockerDesktopClientV0.listContainers

Image Methods

listImages

▸ **listImages**(`options`): `Promise`<`unknown`>

Get the list of images

```
const images = await window.ddClient.listImages();
```

> [!WARNING]
>
> It will be removed in a future version. Use listImages instead.

Parameters

Name	Type	Description
`options`	`never`	(Optional). A JSON like `{ "all": true, "filters": JSON.stringify({ dangling: ["true"] }), "digests": true }` For more information about the different properties see the Docker API endpoint documentation.

Returns

`Promise`<`unknown`>

Inherited from

DockerDesktopClientV0.listImages

Navigation Methods

navigateToContainers

▸ **navigateToContainers**(): `void`

Navigate to the container's window in Docker Desktop.

```
window.ddClient.navigateToContainers();
```

> [!WARNING]
>
> It will be removed in a future version. Use viewContainers instead.

Returns

`void`

Inherited from

DockerDesktopClientV0.navigateToContainers

---

navigateToContainer

▸ **navigateToContainer**(`id`): `Promise`<`any`>

Navigate to the container window in Docker Desktop.

```
await window.ddClient.navigateToContainer(id);
```

> [!WARNING]
>
> It will be removed in a future version.

Parameters

Name	Type	Description
`id`	`string`	The full container id, e.g. `46b57e400d801762e9e115734bf902a2450d89669d85881058a46136520aca28`. You can use the `--no-trunc` flag as part of the `docker ps` command to display the full container id.

Returns

`Promise`<`any`>

A promise that fails if the container doesn't exist.

Inherited from

DockerDesktopClientV0.navigateToContainer

---

navigateToContainerLogs

▸ **navigateToContainerLogs**(`id`): `Promise`<`any`>

Navigate to the container logs window in Docker Desktop.

```
await window.ddClient.navigateToContainerLogs(id);
```

> [!WARNING]
>
> It will be removed in a future version.

Parameters

Name	Type	Description
`id`	`string`	The full container id, e.g. `46b57e400d801762e9e115734bf902a2450d89669d85881058a46136520aca28`. You can use the `--no-trunc` flag as part of the `docker ps` command to display the full container id.

Returns

`Promise`<`any`>

A promise that fails if the container doesn't exist.

Inherited from

DockerDesktopClientV0.navigateToContainerLogs

---

navigateToContainerInspect

▸ **navigateToContainerInspect**(`id`): `Promise`<`any`>

Navigate to the container inspect window in Docker Desktop.

```
await window.ddClient.navigateToContainerInspect(id);
```

> [!WARNING]
>
> It will be removed in a future version.

Parameters

Name	Type	Description
`id`	`string`	The full container id, e.g. `46b57e400d801762e9e115734bf902a2450d89669d85881058a46136520aca28`. You can use the `--no-trunc` flag as part of the `docker ps` command to display the full container id.

Returns

`Promise`<`any`>

A promise that fails if the container doesn't exist.

Inherited from

DockerDesktopClientV0.navigateToContainerInspect

---

navigateToContainerStats

▸ **navigateToContainerStats**(`id`): `Promise`<`any`>

Navigate to the container stats to see the CPU, memory, disk read/write and network I/O usage.

```
await window.ddClient.navigateToContainerStats(id);
```

> [!WARNING]
>
> It will be removed in a future version.

Parameters

Name	Type	Description
`id`	`string`	The full container id, e.g. `46b57e400d801762e9e115734bf902a2450d89669d85881058a46136520aca28`. You can use the `--no-trunc` flag as part of the `docker ps` command to display the full container id.

Returns

`Promise`<`any`>

A promise that fails if the container doesn't exist.

Inherited from

DockerDesktopClientV0.navigateToContainerStats

---

navigateToImages

▸ **navigateToImages**(): `void`

Navigate to the images window in Docker Desktop.

```
await window.ddClient.navigateToImages(id);
```

> [!WARNING]
>
> It will be removed in a future version. Use viewImages instead.

Returns

`void`

Inherited from

DockerDesktopClientV0.navigateToImages

---

navigateToImage

▸ **navigateToImage**(`id`, `tag`): `Promise`<`any`>

Navigate to a specific image referenced by `id` and `tag` in Docker Desktop. In this navigation route you can find the image layers, commands, created time and size.

```
await window.ddClient.navigateToImage(id, tag);
```

> [!WARNING]
>
> It will be removed in a future version. Use viewImage instead.

Parameters

Name	Type	Description

`id`	`string`	The full image id (including sha), e.g. `sha256:34ab3ae068572f4e85c448b4035e6be5e19cc41f69606535cd4d768a63432673`.
`tag`	`string`	The tag of the image, e.g. `latest`, `0.0.1`, etc.

Returns

`Promise`<`any`>

A promise that fails if the container doesn't exist.

Inherited from

DockerDesktopClientV0.navigateToImage

---

navigateToVolumes

▸ **navigateToVolumes**(): `void`

Navigate to the volumes window in Docker Desktop.

```
await window.ddClient.navigateToVolumes();
```

> [!WARNING]
>
> It will be removed in a future version. Use viewVolumes instead.

Returns

`void`

Inherited from

DockerDesktopClientV0.navigateToVolumes

---

navigateToVolume

▸ **navigateToVolume**(`volume`): `void`

Navigate to a specific volume in Docker Desktop.

```
window.ddClient.navigateToVolume(volume);
```

> [!WARNING]
>
> It will be removed in a future version. Use viewVolume instead.

Parameters

Name	Type	Description
`volume`	`string`	The name of the volume, e.g. `my-volume`.

Returns

`void`

Inherited from

DockerDesktopClientV0.navigateToVolume

Other Methods

execHostCmd

▸ **execHostCmd**(`cmd`): `Promise`<`ExecResultV0`>

Invoke a binary on the host. The binary is typically shipped with your extension using the host section in the extension metadata. Note that extensions run with user access rights, this API is not restricted to binaries listed in the host section of the extension metadata (some extensions might install software during user interaction, and invoke newly installed binaries even if not listed in the extension metadata)

```
window.ddClient.execHostCmd(`cliShippedOnHost xxx`).then((cmdResult: any) => {
 console.log(cmdResult);
});
```

> [!WARNING]
>
> It will be removed in a future version. Use exec instead.

Parameters

Name	Type	Description
`cmd`	`string`	The command to be executed.

Returns

`Promise`<`ExecResultV0`>

Inherited from

DockerDesktopClientV0.execHostCmd

---

spawnHostCmd

▸ **spawnHostCmd**(`cmd`, `args`, `callback`): `void`

Invoke an extension binary on your host and get the output stream.

```
window.ddClient.spawnHostCmd(
  `cliShippedOnHost`,
  [`arg1`, `arg2`],
  (data: any, err: any) => {
    console.log(data.stdout, data.stderr);
    // Once the command exits we get the status code
    if (data.code) {
      console.log(data.code);
    }
  }
);
```

> [!WARNING]

> It will be removed in a future version. Use exec instead.

Parameters

Name	Type	Description
`cmd`	`string`	The command to be executed.
`args`	`string`[]	The arguments of the command to execute.
`callback`	(`data`: `any`, `error`: `any`) => `void`	The callback function where to listen from the command output data and errors.

Returns

`void`

Inherited from

DockerDesktopClientV0.spawnHostCmd

---

execDockerCmd

▸ **execDockerCmd**(`cmd`, `...args`): `Promise`<`ExecResultV0`>

You can also directly execute the Docker binary.

```
const output = await window.ddClient.execDockerCmd("info");
```

> [!WARNING]
>
> It will be removed in a future version. Use exec instead.

Parameters

Name	Type	Description
`cmd`	`string`	The command to execute.

Name	Type	Description
`...args`	`string`[]	The arguments of the command to execute.

Returns

`Promise`<`ExecResultV0`>

The result will contain both the standard output and the standard error of the executed command:

```
{
  "stderr": "...",
  "stdout": "..."
}
```

For convenience, the command result object also has methods to easily parse it depending on the output format:

- `output.lines(): string[]` splits output lines.
- `output.parseJsonObject(): any` parses a well-formed JSON output.
- `output.parseJsonLines(): any[]` parses each output line as a JSON object.

If the output of the command is too long, or you need to get the output as a stream you can use the

- spawnDockerCmd function:

```
window.ddClient.spawnDockerCmd("logs", ["-f", "..."], (data, error) => {
  console.log(data.stdout);
});
```

Inherited from

DockerDesktopClientV0.execDockerCmd

---

spawnDockerCmd

▸ **spawnDockerCmd**(`cmd`, `args`, `callback`): `void`

> [!WARNING]

> It will be removed in a future version. Use exec instead.

Parameters

Name	Type
`cmd`	`string`
`args`	`string`[]
`callback`	(`data`: `any`, `error`: `any`) => `void`

Returns

`void`

Inherited from

DockerDesktopClientV0.spawnDockerCmd

---

openExternal

▸ **openExternal**(`url`): `void`

Opens an external URL with the system default browser.

```
window.ddClient.openExternal("https://docker.com");
```

> [!WARNING]
>
> It will be removed in a future version. Use openExternal instead.

Parameters

Name	Type	Description
`url`	`string`	The URL the browser opens (must have the protocol `http` or `https`).

Returns

`void`

Inherited from

DockerDesktopClientV0.openExternal

Toast Methods

toastSuccess

▸ **toastSuccess**(`msg`): `void`

Display a toast message of type success.

```
window.ddClient.toastSuccess("message");
```

> **Warning`**
>
> It will be removed in a future version. Use success instead.

Parameters

Name	Type	Description
`msg`	`string`	The message to display in the toast.

Returns

`void`

Inherited from

DockerDesktopClientV0.toastSuccess

---

toastWarning

▸ **toastWarning**(`msg`): `void`

Display a toast message of type warning.

```
window.ddClient.toastWarning("message");
```

> [!WARNING]
>
> It will be removed in a future version. Use warning instead.

Parameters

Name	Type	Description
`msg`	`string`	The message to display in the toast.

Returns

`void`

Inherited from

DockerDesktopClientV0.toastWarning

---

toastError

▸ **toastError**(`msg`): `void`

Display a toast message of type error.

```
window.ddClient.toastError("message");
```

> [!WARNING]
>
> It will be removed in a future version. Use error instead.

Parameters

Name	Type	Description
`msg`	`string`	The message to display in the toast.

Returns

`void`

Inherited from

DockerDesktopClientV0.toastError

□□□ #2
□ □□□ □□□ 2026-10-31 19:41:29 UTC
□ □□□ □□□ 2026-10-31 19:41:31 UTC