13.23. XR_EXT_permissions_support

Name String

`XR_EXT_permissions_support`

Extension Type

Instance extension

Registered Extension Number

33

Revision

1

Extension and Version Dependencies

- Requires OpenXR 1.0

Last Modified Date

2018-11-05

IP Status

No known IP claims.

Contributors

Mark Young, LunarG
Jules Blok, Epic Jared Cheshier, Pluto VR Nick Whiting, Epic

Overview

OpenXR support spans many platforms and operating systems. Certain functionality that can be used in OpenXR may be readily available on one platform/system, but that same behavior may require user or system permission to be available on another.

Examples of scenarios requiring permission support include:

- Allowing an OpenXR session to access user input even when the session no longer has focus.
- Providing access to OpenXR hardware that normally might be hidden.
- Accessing certain functionality in the underlying operating system that requires user interaction before it is made available (like access to the camera on a smartphone by an external application).

This extension provides the following capabilities:

- The ability to query what permissions are available on the current OpenXR runtime
- The ability to define what permissions the application requires at XrSession create time as well as which permissions are optional.

13.23.1. Permission Names

Permission have names that start with the prefix `OPENXR.permission.`. After the prefix, permission names will have a string that indicates if they are provided as part of the OpenXR core specification or an extension. If they are part of core, then the string `core.` is applied after the prefix. Otherwise, the lower-case extension type (`khr`, `ext`, etc) is applied after the prefix with an additional period ('.') separator inserted after the extension type. The remaining permission name is short but descriptive name for the permission. This short name should always start with a verb and then be followed by one to three word target.

This has the full form of:

```
OPENXR.permission.<core/ext_type>.<short_name>
```

For example, the following might be valid permission names:

- `OPENXR.permission.pluto.allow_unfocused_pose_data`
- `OPENXR.permission.core.enable_eye_tracking`
- `OPENXR.permission.ext.allow_camera_interaction`

13.23.2. Querying Available Permissions

In order to enable the functionality of this extension, you **must** pass the name of the extension into xrCreateInstancevia the XrInstanceCreateInfo `enabledExtensionNames` parameter as indicated in the [extension] section.

Once the extension has been enabled, the list of available permissions may be requested using the xrEnumerateInstancePermissionsEXT command. This command provides information about each available permission including the permission name, a short description, and the permission identifier (or `XrPermissionIdEXT`).

13.23.3. Enabling Permissions

Now that you have a list of available permissions, you enable the permissions by creating the XrSessionCreateInfoPermissionsEXT structure and passing it into the xrCreateSession via the XrSessionCreateInfostructure's `next` parameter. Internal to the `XrSessionCreateInfoPermissionsEXT`, the `requestedPermissionsCount` parameter indicates how many permissions you are requesting as part of your call to `xrCreateSession`. For each of the requesting permissions, you **must** create a XrPermissionRequestEXT structure where you identify the permission identifier and whether or not that indicated permission is optional or required. The `xrCreateSession` command will block as the runtime requests the permissions and only return or resume based on the results of the permission queries. If a permission is not valid, `xrCreateSession` will return `XR_ERROR_PERMISSION_INVALID_EXT`. If all permission requests are permitted by the system/user, `xrCreateSession` will return `XR_SUCCESS`. If a permission is denied, the return code will depend on whether or not the requested permission was labeled as optional by the application. If a permission is optional but denied access, `xrCreateSession` will return `XR_OPTIONAL_PERMISSION_UNAVAILABLE_EXT`. This return value is not considered an error since the permission was optional. However, if the permission was marked as required and denied access, `XR_ERROR_REQUIRED_PERMISSION_UNAVAILABLE_EXT` will be returned.

New Object Types

```
XrPermissionIdEXT
```

New Flag Types

None

New Enum Constants

XrStructureType enumeration is extended with:

```
XR_TYPE_PERMISSION_PROPERTIES_EXT
XR_TYPE_PERMISSION_REQUEST_EXT
XR_TYPE_SESSION_CREATE_INFO_PERMISSIONS_EXT
```

XrResult enumeration is extended with:

- New success codes:

```
XR_OPTIONAL_PERMISSION_UNAVAILABLE_EXT
```

- New error codes:

```
XR_ERROR_PERMISSION_INVALID_EXT
XR_ERROR_REQUIRED_PERMISSION_UNAVAILABLE_EXT
```

New Enums

None

New Structures

```
typedef struct XrPermissionPropertiesEXT {
    XrStructureType             type;
    const void* XR_MAY_ALIAS    next;
    char                        permissionName[128];
    char                        permissionDescription[256];
    XrPermissionIdEXT           permissionId;
} XrPermissionPropertiesEXT;
```

Member Descriptions

- `type` is the type of this structure.
- `next` is `NULL` or a pointer to an extension-specific structure.
- `permissionName` is the name of the permission.
- `permissionDescription` is a short description of the functionality provided by the permission.
- `permissionId` is a unique value for the permission request. This value may vary by process, but should at least be unique per process.

Valid Usage (Implicit)

- `type` **must** be `XR_TYPE_PERMISSION_PROPERTIES_EXT`
- `next` **must** be `NULL`

```
typedef struct XrPermissionRequestEXT {
    XrStructureType             type;
    const void* XR_MAY_ALIAS    next;
    XrPermissionIdEXT           permissionId;
    XrBool32                    optional;
} XrPermissionRequestEXT;
```

Member Descriptions

- `type` is the type of this structure.
- `next` is `NULL` or a pointer to an extension-specific structure.
- `permissionId` is the unique value for the permission request returned in the XrPermissionPropertiesEXTstructure.
- `optional` is an `XrBool32` value that indicates if the permission is optional (XR_TRUE) or required (XR_FALSE).

Valid Usage (Implicit)

- `type` **must** be `XR_TYPE_PERMISSION_REQUEST_EXT`
- `next` **must** be `NULL`

```
typedef struct XrSessionCreateInfoPermissionsEXT {
    XrStructureType                 type;
    const void* XR_MAY_ALIAS        next;
    uint32_t                        requestedPermissionsCount;
    const XrPermissionRequestEXT*   requestedPermissions;
} XrSessionCreateInfoPermissionsEXT;
```

Member Descriptions

- `type` is the type of this structure.
- `next` is `NULL` or a pointer to an extension-specific structure.
- `requestedPermissionsCount` is the number of `requestedPermissions` being supplied to the runtime by the application.
- `requestedPermissions` is a pointer to an array of size `requestedPermissionsCount` of XrPermissionRequestEXT structures containing the information on the permissions being requested.

Valid Usage (Implicit)

- `type` **must** be `XR_TYPE_SESSION_CREATE_INFO_PERMISSIONS_EXT`
- `next` **must** be `NULL`
- If `requestedPermissionsCount` is not `0`, `requestedPermissions` **must** be a pointer to an array of `requestedPermissionsCount` valid `XrPermissionRequestEXT` structures

New Functions

```
XrResult xrEnumerateInstancePermissionsEXT(
    XrInstance                                  instance,
    uint32_t                                    propertyCapacityInput,
    uint32_t*                                   propertyCountOutput,
    XrPermissionPropertiesEXT*                  properties);
```

Parameter Descriptions

- `instance` is the instance that the object was created under.
- `propertyCapacityInput` is the capacity of the properties array, or 0 to indicate a request to retrieve the required capacity.
- `propertyCountOutput` is a pointer to the count of properties written, or a pointer to the required capacity in the case that `propertyCapacityInput` is 0.
- `properties` is a pointer to an array of XrPermissionPropertiesEXT structures, but **can** be `NULL` if `propertyCapacityInput` is 0.
- See "Buffer Size Parameters" chapter for a detailed descriptions of retrieving the required `properties` size.

Valid Usage (Implicit)

- `instance` **must** be a valid `XrInstance` handle
- If `propertyCountOutput` is not `NULL`, `propertyCountOutput` **must** be a pointer to a `uint32_t` value
- If `propertyCapacityInput` is not `0`, `properties` **must** be a pointer to an array of `propertyCapacityInput` `XrPermissionPropertiesEXT` structures

Return Codes

Success

- `XR_SUCCESS`

Failure

- `XR_ERROR_INSTANCE_LOST`
- `XR_ERROR_RUNTIME_FAILURE`
- `XR_ERROR_SIZE_INSUFFICIENT`
- `XR_ERROR_VALIDATION_FAILURE`

New Function Pointers

None

Issues

None

Version History

- Revision 1, 2018-11-05 (Mark Young)
  - Initial draft