ME Bridge

Picture

The ME Bridge is able to interact with Applied Energistics 2. You can retrieve items, craft items, get all items as a list and more. The ME Bridge uses one channel.

Requirement

Requires the Applied Energistics 2 mod to be installed

Peripheral Name	Interfaces with	Has events	Introduced in

Peripheral Name	Interfaces with	Has events	Introduced in
meBridge	ME System	Yes	0.3b

Failure

You need to place the inventory/tank you want to use to export/import stuff next to the ME Bridge and **NOT** next to the computer!

1.20.1-0.7 and older

Info

If you are playing on 0.7 for the minecraft version 1.21.1 and newer, please use the content tab for "1.21.1-0.7 and newer". The following documentation contains the legacy ME and RS bridge features which will be replaced with the next stable AP version.

The new ME and RS Bridge systems will eventually be back ported to 1.20.1 and 1.19.2 with 0.8.

Events

crafting

Fires when a crafting job starts or fails.
Values: 1. `success: boolean` Indicates whether a crafting job has successfully started or not 2. `message: string` A message about the status of the crafting job
These values are equivalent to the return values of `craftItem()`.

```
local event, success, message = os.pullEvent("crafting")
if success then
    print("A crafting job has successfully started")
else
    print("A crafting job has failed to start")
end
```

Warning

The `crafting` event will fire when the ME Bridge is connected to an active ME System that is performing crafting operations. These operations **need** to be started by the bridge itself!

Tip

You can use the command `/advancedperipherals getHashItem` with an item in your hand to get the MD5 hash of the NBT tags of the item. An MD5 Hash can look like this `ae70053c97f877de546b0248b9ddf525`.

Functions

Info

The item arguments(`item: table`) accepts our item filters, you can check the syntax of these filters here.

craftItem

```
craftItem(item: table[, craftingCpu: string]) -> boolean, err: string
```

Tries to craft the provided `item`. If a `craftingCpu` 's name is provided then it will use that cpu to craft the `item`.

Item Properties

Check the item filters guide for more info!

item	Description
name: `string`	The registry name of the item or a tag
count: `number?`	The amount of the item to craft

item	Description
nbt: `string?`	NBT to match the item on

OR

item	Description
fingerprint: `string`	A unique fingerprint which identifies the item to craft
count: `number?`	The amount of the item to craft

---

craftFluid

```
craftFluid(fluid: table[, craftingCpu: string]) -> boolean, err: string
```

Tries to craft the provided `fluid`. If a `craftingCpu`'s name is provided then it will use that cpu to craft the `fluid`.

Fluid Properties

Check the fluid filters guide for more info!

fluid	Description
name: `string`	The registry name of the fluid or a tag
count: `number?`	The amount of the fluid to craft
nbt: `string?`	NBT to match the fluid on

OR

fluid	Description

fluid	Description
fingerprint: `string`	A unique fingerprint which identifies the fluid to craft
count: `number?`	The amount of the fluid to craft
---	

getItem

```
getItem(item: table) -> table, err: string
```

Returns a table with information about the item type in the system.

Properties

result	Description
name: `string`	The registry name of the item
fingerprint: `string?`	A unique fingerprint which identifies the item to craft
amount: `number`	The amount of the item in the system
displayName: `string`	The display name for the item
isCraftable: `boolean`	Whether the item has a crafting pattern or not
nbt: `string?`	NBT to match the item on
tags: `table`	A list of all of the item tags

importItem

```
importItem(item: table, direction: string) -> number, err: string
```

Imports an `item` from a container in the `direction` to the ME System.
Returns the number of the `item` imported into the system.

Since version 0.7r

You can now use both relative (`right`, `left`, `front`, `back`, `top`, `bottom`) and cardinal (`north`, `south`, `east`, `west`, `up`, `down`) directions for the `direction` argument.

```
local bridge = peripheral.find("meBridge")

-- Imports 32 dirt from the container above into the system
bridge.importItem({name="minecraft:dirt", count=1}, "up")
```

---

exportItem

```
exportItem(item: table, direction: string) -> number, err: string
```

Exports an `item` to a container in the `direction` from the ME bridge block.
Returns the number of the `item` exported into the container.

```
local bridge = peripheral.find("meBridge")

-- Exports 1 "Protection I" book into the container above
bridge.exportItem({name="minecraft:enchanted_book", count=1,
nbt="ae70053c97f877de546b0248b9ddf525"}, "up")
```

---

importItemFromPeripheral

```
importItemFromPeripheral(item: table, container: string) -> number, err: string
```

Similar to `importItem()` it imports an `item` from a container which is connected to the peripheral network.
`container` should be the exact name of the container peripheral on the network.
Returns the number of the `item` imported from the container.

---

exportItemToPeripheral

```
exportItemToPeripheral(item: table, container: string) -> number, err: string
```

Similar to `exportItem()` it exports an `item` to a container which is connected to the peripheral network.
`container` should be the exact name of the container peripheral on the network.
Returns the number of the `item` exported into the container.

---

getEnergyStorage

```
getEnergyStorage() -> number, err: string
```

Returns the stored energy of the whole ME System in AE.

---

getMaxEnergyStorage

```
getMaxEnergyStorage() -> number, err: string
```

Returns the maximum energy storage capacity of the whole ME system in AE.

---

getEnergyUsage

```
getEnergyUsage() -> number, err: string
```

Returns the energy usage of the whole ME System in AE/t.

---

getCraftingCPUs

```
getCraftingCPUs() -> table, err: string
```

Returns a list of all connected crafting cpus.

CPU Properties

cpu	Description
storage: `number`	The amount of storage the CPU has
coProcessors: `number`	The number of coprocessors the CPU has
isBusy: `boolean`	If the cpu is currently crafting

isItemCrafting

```
isItemCrafting(item: table[, craftingCpu: string]) -> boolean, err: string
```

Returns true if a crafting job for the `item` exists. If a `craftingCpu`'s name is provided then it will check only if that cpu is crafting the `item`.

isItemCraftable

```
isItemCraftable(item: table) -> boolean, err: string
```

Returns true if the `item` is craftable.

listCraftableItems

```
listCraftableItems() -> table, err: string
```

Returns a list of information about all craftable items

Properties

item / fluid	Description
name: `string`	The registry name of the item

item / fluid	Description
fingerprint: `string?`	A unique fingerprint which identifies the item to craft
amount: `number`	The amount of the item in the system
displayName: `string`	The display name for the item
isCraftable: `boolean`	Whether the item has a crafting pattern or not
nbt: `string?`	NBT to match the item on
tags: `table`	A list of all of the item tags

```lua
local bridge = peripheral.find("meBridge")

-- print out all craftable items
craftableItems = bridge.listCraftableItems()
for _, item in pairs(craftableItems) do
    print(item.name)
end
```

listCraftableFluid

```
listCraftableFluid() -> table, err: string
```

Returns a list of information about all craftable fluids

listItems

```
listItems() -> table, err: string
```

Returns a list of information about all items in the ME System.

listFluid

```
listFluid() -> table, err: string
```

Returns a list of information about all fluids in the ME System.

listGas

```
listGas() -> table, err: string
```

Returns a list of information about all gases (Applied Mekanistics) in the ME System.

Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

listCells

```
listCells() -> table, err: string
```

Returns a list of information about all cells in the disk drives of the ME System.

cell	Description
item: `string`	The name of the cell. e.g. `ae2:64k_storage_cell
cellType: `string`	The type of the cell. `item` or `fluid`
bytesPerType: `int`	The bytes per type
totalBytes: `int`	Total available bytes of the cell

Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

getTotalItemStorage

```
getTotalItemStorage() -> int, err: string
```

Returns how much total item storage the system offers

Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

getTotalFluidStorage

```
getTotalFluidStorage() -> int, err: string
```

Returns how much total fluid storage the system offers

Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

getUsedItemStorage

```
getUsedItemStorage() -> int, err: string
```

Returns how much item storage is used

Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

getUsedFluidStorage

```
getUsedFluidStorage() -> int, err: string
```

Returns how much fluid storage is used

Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

getAvailableItemStorage

```
getAvailableItemStorage() -> int, err: string
```

1.21.1-0.7 and newer

Returns how much item storage is available

Info
Added in version 1.18.2-0.7.24r | 1.19.2-0.7.23b

If you are playing on 0.7 for the minecraft version 1.20.1 and older, please use the content tab for "1.20.1-0.7 and older". The following documentation also works for the canary 0.8 version.

The new ME and RS Bridge systems will eventually be back ported to 1.20.1 and 1.19.2 with 0.8.

```
getAvailableFluidStorage() -> int, err: string
```

Returns how much fluid storage is available

Functionality

The RS and ME Bridge now share the same functionality. Check this Guide for the whole documentation for every available feature.

Examples

Requester

The requester is a successor for the old "Automatic Autocrafting" script which uses the old ME Bridge functions. It currently only works with AP on 1.21.1 or on 1.19.2 0.8 and supports both the ME and the RS Bridge at the same time.

This script automatically schedules crafting jobs for pre-defined items, fluids and mekanism chemicals Do you want 500 glass in your me system at all times? Add glass to the list and the script will craft it for you. No need for level emitters or crafting cards!

You can find instructions on how to install the script here

Automatic Autocrafting

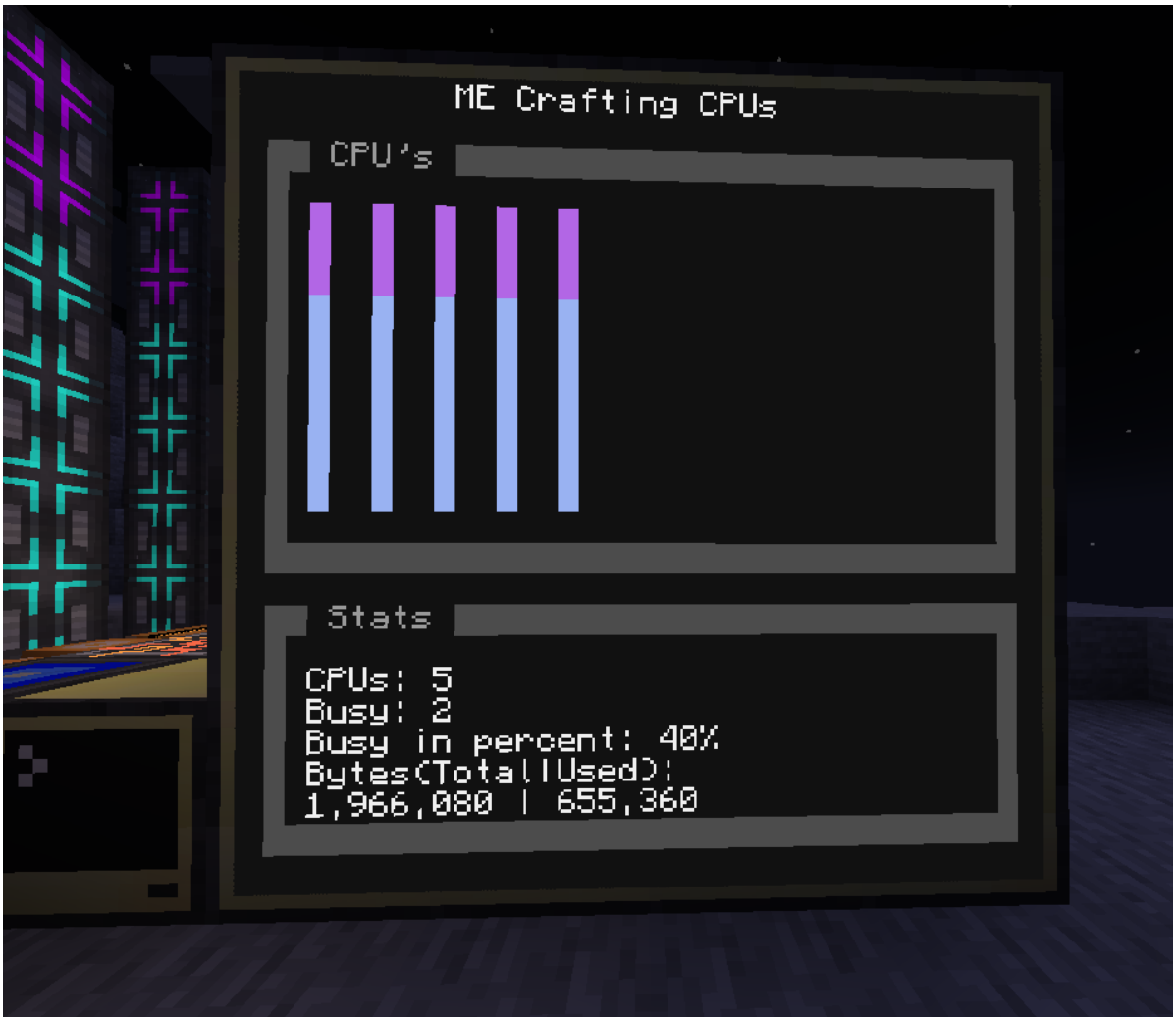
This script automatically crafts items in a list. Do you want 500 glass in your me system at all times? Add glass to the list and the script will craft it for you. No need for level emitters or crafting cards!

You can find instructions on how to install the script here

ME Crafting CPUs

This script shows you some statistics about the ME crafting cpus.

You can find instructions on how to install the script here



Changelog/Trivia

0.7r

The ME Bridge does uses computercraft relative and cardinal directions. We also changed some function names.

0.4b

Reworked the system of the ME Bridge, it now has more features and a new system for the `item`

parameter.

0.3.9b

Added the `importItem` and `exportItem` from container functions.

0.3b

Added the ME Bridge with a good amount of features.

August 3, 2025 April 17, 2021 GitHub   +6