# Draw Steel Macros

These macros were inspired by conversations on the MCDM discord in the Foundry VTT System channel. Thanks to the contributors there for their suggestions, corrections, and testing.

My thanks to @ChaosOS for his assistance in answering my questions, writing key parts of these macros, and for generally improving them. They are much cleaner now than their first iteration. Any remaining errors are my own.

Thanks to @Drafty for the icons used in the surge macros.

Thanks to @TiTo for their work on the Set Temp Stamina macro.

Thanks to @Zhell and @Izzy for their macros involving token ring colors and combat initiative groups.

## Macro Permissions

Macros created in the standard Foundry macro panel on the sidebar may not allow players to use them by default. Right click on the macro and use the Configure Ownership panel to edit their access. I believe that Observer access is sufficient, though I am not certain of all the nuances of that.

If you put your macros in a compendium, you probably need to give the players Observer access to the compendium by using the Configure Ownership panel for the compendium. It appears that macros in the compendium inherit their permissions from the compendium itself.

## Useful Resources

- Macro-Polo channel in the Foundry Discord.
- Flix's Guides to Macros

## Wish List

Requests for macros that I haven't had a chance to look at yet:

- Implement a Chosen One (complication) macro. From KJTailor in discord.
- Create a standard template of empty combat groups. From Huge_Fanboy in discord.
- Allow the damage macro to apply to selected target if there is one (or more), or default to owned character otherwise; include flag at top to turn this on or off if someone wants it to always work on owned only, or selected only. Note: How could/should this interact with Draw Steel Quick Strike module since it enables and prioritizes using targeted tokens for applying damage, healing, effects, etc instead of selected ones.
- Macros to easily move selected tokens between combat groups, remove them, etc. From KJTailor in discord.
- Macros to update from compendium more efficiently.

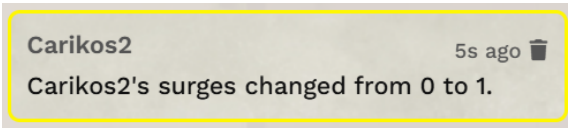
# Macros for Surges

The following macros are useful for adding and removing surges, and spending them to apply additional damage.

## +1/-1 Surge

This macro will increment the select actors' surges by +1 when clicked.  If you hold CTRL while clicking, it will instead subtract 1.

If you have multiple tokens selected, this will apply to every selected hero token while skipping monster tokens.




```
// This macro will increment your surges by +1 when clicked.  If you hold CTRL while clicking, it will instead subtract 1.

// Detect if CTRL is held down
const isCtrl = game.keyboard.downKeys.has("ControlLeft") || game.keyboard.downKeys.has("ControlRight");

const actors = canvas.tokens.controlled.map((t) => t.actor); // update all selected tokens

actors
  .filter((actor) => actor.type === "hero")
  .forEach(async (actor) => {

    let oldVal = actor.system.hero.surges
    let newVal = Math.max(actor.system.hero.surges + (isCtrl ? -1 : 1), 0);

    await actor.update({ 'system.hero.surges': newVal })

    ChatMessage.create({
      author: game.user,
      speaker: ChatMessage.getSpeaker({ actor }),
      content: `${actor.name}'s surges changed from ${oldVal} to ${newVal}.`
    });
});
```

## Use Surges (Dialog Box)

This macro will use a popup dialog box to prompt you for how many surges to spend.

You can also specify any surges that you should gain before spending them on this action.  For example, if you are spending a hero token or using an ability that provides bonus surges for

*Popup Dialog:*

immediate use (ex: Primordial Strike), you could add +1 or +2 surges in addition to specifying how many you are spending.

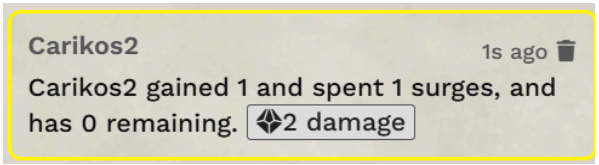
The macro will update the actor's surge total and send either a damage enricher or a damage roll to the chat window.

The first variable in the macro, called output, lets you specify if you want the macro to output a damage enricher or just the final damage roll result.

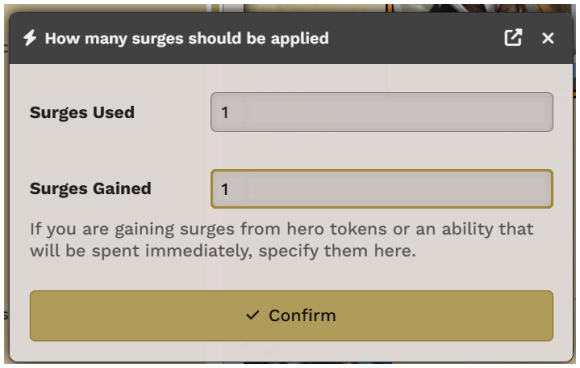
If you use a damage enricher, you will need to click the enricher button and then also apply the damage.

If you set the output variable to "damage" it will skip the enricher message and just go directly to the Apply Damage button.

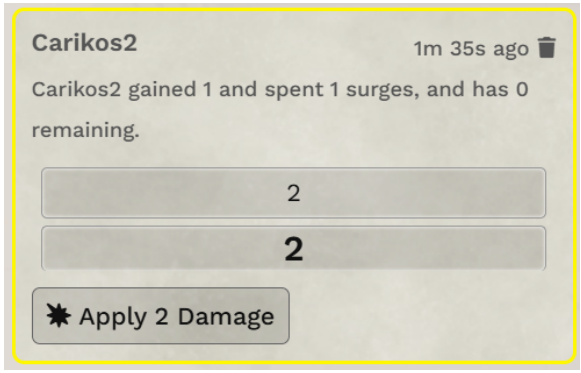
*Damage Enricher Output:*




Thanks to @ChaosOS for his help on this macro!




*Damage Roll Output:*




```
// This macro will prompt you for the number of surges to spend.  It will also
// allow you to specify that you are gaining some additional surges prior to using
// them, whether from spending hero tokens or from an ability that grants a surge
// for immediate use.

// Specify if you want the output to use a damage enricher or just directly go to
// the Apply Damage button.
// Choose one of the following and comment out the other.
// let output = "enricher";
let output = "damage";

if (!actor) {
  ui.notifications.warn("Please select your token first.");
  return;
}
```

```
const {createFormGroup, createNumberInput} = foundry.applications.fields;

const content = document.createElement("div");

const surgesUsed = createFormGroup({
    label: "Surges Used",
    rootId: "surges",
    input: createNumberInput({ name: "spent" })
})

const surgesGained = createFormGroup({
    label: "Surges Gained",
    hint: "If you are gaining surges from hero tokens or an ability that will be spent immediately, specify them here.",
    rootId: "surges",
    input: createNumberInput({ name: "gained" })
})

content.append(surgesUsed, surgesGained)

const fd = await ds.applications.api.DSDialog.input({
    content,
    window: {
        title: "How many surges should be applied",
        icon: "fa-solid fa-bolt"
    }
})

if (!fd) return;

if (fd.gained === null) {fd.gained = 0;}
if (fd.spent === null) {fd.spent = 0;}

const oldVal = actor.system.hero.surges;

if ((oldVal + fd.gained) < fd.spent) {
    ui.notifications.warn("You do not have enough surges");
    return;
}

const newVal = Math.max(oldVal + fd.gained - fd.spent, 0);

await actor.update({ 'system.hero.surges': newVal })

const rollData = actor.getRollData();
```

```
const dmg = fd.spent * Math.max(rollData.M, rollData.A, rollData.R, rollData.I, rollData.P);

// OPTIONAL OUTPUT FORMATS - choose one of the following by setting the output variable at the top of the macro.

if ( output == 'enricher') {
  ChatMessage.create({
      author: game.user,
      speaker: ChatMessage.getSpeaker({ token }),
      content: `<p>${token.name} gained ${fd.gained} and spent ${fd.spent} surges, and has ${newVal} remaining. [[/damage ${dmg}]]{${dmg} damage}.</p>`
  });
} else {
  const roll = new ds.rolls.DamageRoll(dmg.toString());
  roll.toMessage({
       author: game.user,
       speaker: ChatMessage.getSpeaker({ token }),
       flavor: `<p>${token.name} gained ${fd.gained} and spent ${fd.spent} surges, and has ${newVal} remaining.</p>`,
       rollMode: "roll"
      });
}
```

## Use 1 Surge (No Dialog)

This is a simpler macro that just uses one surge and sends either a damage roll or a damage enricher to chat.

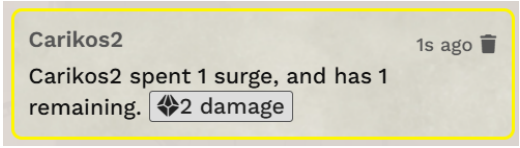
You can alter the number of surges by editing the first variable at the top of the macro, if you want to have a copy of the macro for 2 surges or 3 surges.

The second variable, called output, lets you specify if you want the macro to output a damage enricher or just the final damage roll result.

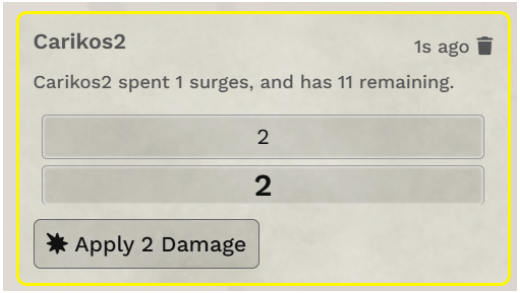
If you use a damage enricher, you will need to click the enricher button and then also roll the damage.

If you set the output variable to "damage" it will skip the enricher message and just go directly to the Apply Damage button.

**Output = Enricher**:




**Output = Damage**:




```
// Spend one surge and use send a damage enricher to the chat window.
// Set the number of surges you want this macro to use in the first variable.

// set the number of surges to use:
let surgesUsed = 1;

// Specify if you want the output to use a damage enricher or just the Apply Damage button.
// Choose one of the following and comment out the other.
//let output = "enricher";
let output = "damage";

//-------------------------------------------------------------------
if (!actor) {
  ui.notifications.warn("Please select your token first.");
  return;
}

let oldVal = actor.system.hero.surges
if (oldVal < surgesUsed) {
  ui.notifications.warn("You do not have enough surges.");
  return;
}

const newVal = Math.max(actor.system.hero.surges - surgesUsed, 0);
```

```
await actor.update({ 'system.hero.surges': newVal })

const rollData = actor.getRollData();

const dmg = surgesUsed * Math.max(rollData.M, rollData.A, rollData.R, rollData.I, rollData.P);

// OPTIONAL OUTPUT FORMATS - choose one of the following by setting the output variable at the top of the macro.

if ( output == 'enricher') {
  ChatMessage.create({
    author: game.user,
    speaker: ChatMessage.getSpeaker({ token }),
    content: `${token.name} spent ${surgesUsed} surges, and has ${newVal} remaining. [[/damage ${dmg}]]{${dmg} damage }`
  });
} else {
  const roll = new ds.rolls.DamageRoll(dmg.toString());
  roll.toMessage({
      author: game.user,
      speaker: ChatMessage.getSpeaker({ token }),
      flavor: `${token.name} spent ${surgesUsed} surges, and has ${newVal} remaining.`
  });
}
```

## Macros for Other Resources

In addition to macros, there is also a module that gives a panel that lets you gain, spend and track heroic resources (as well as surges), including the additional HR gained from triggers mid-round.  As of March 2026 it supports the core classes, but may be expanded to allow custom classes in the future.  See the Resources UI Module discord post by Cora or the github repository.

### +1/-1 Heroic Resource

This macro will increment the select actors' heroic resources by +1 when clicked.  If you hold CTRL while clicking, it will instead subtract 1.

Classes with Clarity for their heroic resource can go into negative values, but everyone else is stopped at 0.

If you have multiple tokens selected, this will apply to every selected hero token while skipping monster tokens.

```
// This macro will increment your heroic resource by +1 when clicked.  If you hold CTRL while clicking, it will instead subtract 1.

// Detect if CTRL is held down
const isCtrl = game.keyboard.downKeys.has("ControlLeft") || game.keyboard.downKeys.has("ControlRight");

let newVal = 0;
const actors = canvas.tokens.controlled.map((t) => t.actor); // update all selected tokens

actors
  .filter((actor) => actor.type === "hero")
  .forEach(async (actor) => {
    let oldVal = actor.system.hero.primary.value
    if (actor.system.hero.primary.label == "Clarity") {
      // allow Talents to go into negative Clarity
      newVal = actor.system.hero.primary.value + (isCtrl ? -1 : 1);
    } else {
      newVal = Math.max(actor.system.hero.primary.value + (isCtrl ? -1 : 1),0);
    }
    await actor.update( { 'system.hero.primary.value': newVal });

    ChatMessage.create({
      author: game.user,
      speaker: ChatMessage.getSpeaker({ actor }),
      content: `${actor.name}'s Heroic Resource changed from ${oldVal} to ${newVal}.`
    });
});
```

If you would prefer for the macro to work for players only based on their assigned character rather than which tokens are selected, there was a modified version in discord by ChaosOS that determines the actor based on the user's assigned character:

```
// This macro will increment a player's preassigned character's heroic resource by +1 when clicked.  If you hold CTRL while clicking, it will instead subtract 1.

// Detect if CTRL is held down
```

```
const isCtrl = game.keyboard.downKeys.has("ControlLeft") || game.keyboard.downKeys.has("ControlRight");

let newVal = 0;

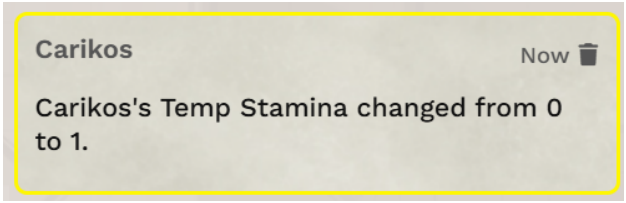
actor = game.user.character

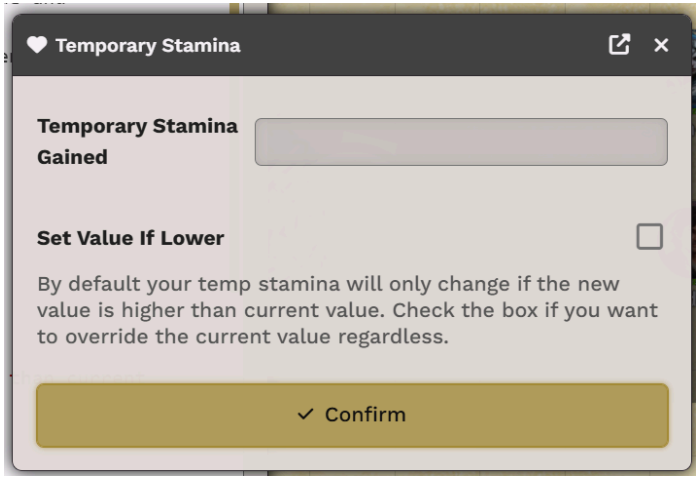
let oldVal = actor.system.hero.primary.value
if (actor.system.hero.primary.label == "Clarity") {
  // allow Talents to go into negative Clarity
  newVal = actor.system.hero.primary.value + (isCtrl ? -1 : 1);
} else {
  newVal = Math.max(actor.system.hero.primary.value + (isCtrl ? -1 : 1),0);
}
await actor.update( { 'system.hero.primary.value': newVal });

ChatMessage.create({
  author: game.user,
  speaker: ChatMessage.getSpeaker({ actor }),
  content: `${actor.name}'s Heroic Resource changed from ${oldVal} to ${newVal}.`
});
```

## Set Temp Stamina

This macro will let you set the temporary stamina value on any selected PC tokens. It will prompt you for the new value. By default, if the new value is lower than the current temporary stamina, the PC will not be updated and a chat message will indicate that nothing changed. If the new value is higher, then it will update the token and indicate the old and new values in chat.
If you wish to truly override the temporary stamina and set a specific value even if it is lower than the current, check the checkbox on the form and the entered value will be applied (minimum of 0).




Thanks to @TiTo for their contributions to this macro!

```
// This macro will prompt you for the new number of temporary stamina to apply to the
// selected tokens.  By default it will compare the new value to the existing value and
// only update it if the new value is higher.  There is a checkbox that will let
// you disable that check and assign any value, even if it is lower than the current.

const {createFormGroup, createNumberInput} = foundry.applications.fields;
const content = document.createElement("div");
const tempstamGained = createFormGroup({
    label: "Temporary Stamina Gained",
    rootId: "tempstam",
    input: createNumberInput({ name: "gained" })
})
const override_form = createFormGroup({
  label: "Set Value If Lower",
  hint: "By default your temp stamina will only change if the new value is higher than current value.  Check the box if you want to override the current value
regardless.",
  rootId: "override",
  input: foundry.applications.fields.createCheckboxInput({ name: "override" })
})
content.append(tempstamGained, override_form);

const fd = await ds.applications.api.DSDialog.input({
    content,
    window: {
        title: "Temporary Stamina",
        icon: "fa-solid fa-heart"
    }
})
if (!fd) return;

if (fd.gained === null) {fd.gained = 0;}

let newVal = 0;
let contentMsg = "";

const actors = canvas.tokens.controlled.map((t) => t.actor); // update all selected tokens
actors
  .filter((actor) => actor.type === "hero")
  .forEach(async (actor) => {
  const oldVal = actor.system.stamina.temporary;
    if (fd.override) {
      newVal = Math.max(fd.gained,0); // don't compare to existing value, just apply it; min of 0
    } else {
      newVal = Math.max(oldVal, fd.gained, 0);  // only update if greater
    }
```

```
  if (newVal == oldVal) {
    contentMsg = `<p>${actor.name} keeps ${oldVal} temp stamina, instead of ${fd.gained}.</p>`
  } else {
    contentMsg = `<p>${actor.name}'s Temp Stamina changed from ${oldVal} to ${newVal}.</p>`;
  }
  ChatMessage.create({
    author: game.user,
    speaker: ChatMessage.getSpeaker({ actor }),
    content: contentMsg
  });
  const data = {
    "system.stamina.temporary": newVal
  };
  await actor.update(data);
});
```

## Set Temp Stamina (Allow Players to Affect Targets)

The normal macro above must be run by someone who owns and selects the tokens who are getting the temp stamina. If you want a player to be able to give temp stamina to the other PCs, there is a security limitation. You must either have the player as an owner on all of the other PC tokens, or you need a way to bypass the normal Foundry security to let the player affect other tokens. The second approach requires a few steps:

1. **Install the Advanced Macros module**. Links: Foundry Github This module lets players run a macro on the Director/GM's session. However, it also uses the Director's selected/targeted tokens, not the tokens targeted by the player running the macro, so we need a two-step process. The first macro the PC run using their own session will identify the PC's targets and get their uuids, then it will call the second macro that runs as the Director/GM and uses the token uuids sent to it in the parameters.

2. **Create the GM/Director version of the macro**:

   Create a new macro, call it "Grant Temp Stamina (GM)" (if you want a different name, you'll need to update it in the 2nd macro below).

   Set the macro type to Script.

   If you have Advanced Macros installed, you'll see a new drop-down in the macro editor that is labeled "Execute for Specific User". Set this to Director.

   Copy/paste the code below into the macro's main body and then save it.

   Right click on the macro in the sidebar and Configure Ownership to have all players (or selected players) to have Observer permissions to this macro.

   **Grant Temp Stamina (GM)**

```
// Requires the Advanced Macros module
// Execute for Specific User should be set to Director.  It will run
// on the Director's session so they must be logged in.
// Players must have Observer ownership for both this macro and
// the "Grant Temp Stamina (PC)" macro.
```

```
const amount = gained ?? 2;

for (const uuid of targetUuids ?? []) {
  const tokenDoc = await fromUuid(uuid);
  const actor = tokenDoc?.actor;
  if (!actor || actor.type !== "hero") continue;

  const oldVal = actor.system.stamina.temporary;
  const newVal = Math.max(oldVal, amount, 0);   // only raise, never lower

  await ChatMessage.create({
    author: userId ?? game.user.id,
    speaker: ChatMessage.getSpeaker({ actor }),
    content: newVal === oldVal
      ? `<p>${actor.name} keeps ${oldVal} temp stamina, instead of ${amount}.</p>`
      : `<p>${actor.name}'s Temp Stamina changed from ${oldVal} to ${newVal}.</p>`,
  });

  await actor.update({ "system.stamina.temporary": newVal });
}
```

3. **Create the PC version of the macro:**

   Create a new macro, call it whatever you want - I'm using "Grant Temp Stamina (PC)".

   Set the macro type to Script.

   Leave Execute for Specific User blank - no option selected.

   Copy/paste the code below into the macro's main body and then save it.

   Right click on the macro in the sidebar and Configure Ownership to have all players (or selected players) to have Observer permissions to this macro.

   **Grant Temp Stamina (PC)**

```
// Requires the Advanced Macros module
// Execute for Specific User should be left blank.
// Players must have Observer ownership for both this macro and
// the "Grant Temp Stamina (GM)" macro.

const targetUuids = Array.from(game.user.targets).map((t) => t.document.uuid);

if (!targetUuids.length) return ui.notifications.warn("No tokens targeted.");
if (!game.users.activeGM) return ui.notifications.warn("No GM is logged in — cannot apply temp stamina.");

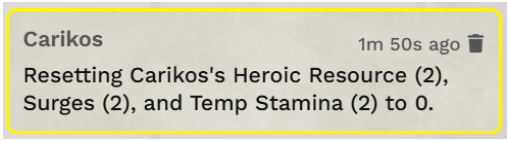
const gmMacro = game.macros.getName("Grant Temp Stamina (GM)");
if (!gmMacro) return ui.notifications.error("GM-side macro not found or not shared with you.");
```

```
await gmMacro.execute({ targetUuids, gained: 2, userId: game.user.id });
```

4. Players will target the tokens they want to affect, then use the "Grant Temp Stamina (PC)" macro.  The Director must be logged in.

## Reset Resources to 0

This macro can be used at the end of encounters to reset the temporary stamina, surges and heroic resources of all selected tokens to zero.




```
// This macro will reset all selected PC tokens to 0 for their temporary stamina, surges, and heroic resources.

const actors = canvas.tokens.controlled.map((t) => t.actor); // update all selected tokens
actors
  .filter((actor) => actor.type === "hero")
  .forEach(async (actor) => {

    ChatMessage.create({
      author: game.user,
      speaker: ChatMessage.getSpeaker({ actor }),
      content: `Resetting ${actor.name}'s Heroic Resource (${actor.system.hero.primary.value}), Surges (${actor.system.hero.surges}), and Temp Stamina
(${actor.system.stamina.temporary}) to 0.`
    });

    const data = {
      "system.hero.primary.value": 0,
      "system.hero.surges": 0,
      "system.stamina.temporary": 0
    };
    await actor.update(data);
});
```

## Set Hero Tokens To Logged In Players

This macro was provided by ChaosOS and Huge_Fanboy on the discord.  It resets the number of hero tokens to the number of non-GM players who are currently logged in, instead of the default reset in the system UI which is based on the number of configured players regardless of their logged in status.

```
let oldVal = game.settings.get('draw-steel', 'heroTokens').value
const nonGM = game.users.filter(u => !u.isGM && u.active)
await game.settings.set('draw-steel', 'heroTokens', { value: nonGM.length })
let newVal = game.settings.get('draw-steel', 'heroTokens').value

//Chat message to show the new value of Hero Tokens

    ChatMessage.create({
      content: `The Director reset Hero Tokens from ${oldVal} to ${newVal}.`
    });
```

# Damage Macros

## 1d6+Level Damage

This macro will roll a d6+level damage and send it to the chat panel with an Apply Damage button.

UPDATE:  If you want the macro to actually apply the damage to the selected token automatically, without needing to also click the button in chat, set the AutoApply variable to ***true*** at the top of the macro.  If you prefer to use the chat button to apply the damage, set the AutoApply variable to ***false***.

UPDATES:

2025-11-17:  Fixed a bug on the takeDamage() function call that was not correctly applying damage types.

```
// Roll 1d6+Level damage to the chat window.

// If you want the macro to automatically apply the damage to the selected token,
// change the AutoApply variable to true instead of false.
let AutoApply = true;

if (AutoApply) {
  autoApplyPrefix = "Automatically applying";
  autoApplySuffix = "<br>Do not use the Apply button except to reapply.";
} else {
  autoApplyPrefix = "Apply";
  autoApplySuffix = "Select a token and hit the Apply button below.";
}

if (!actor) {
  ui.notifications.warn("Please select your token first.");
  return;
}

const lvl = actor.system.level;
const roll = new ds.rolls.DamageRoll('1d6+'+lvl);

await roll.toMessage({
  speaker: ChatMessage.getSpeaker({ token }),
  flavor: `<p>${autoApplyPrefix} ${roll.formula} ${roll.type} damage. ${autoApplySuffix}</p>`,
  rollMode: "roll"
});

// auto apply damage to token and report the change in stamina if this option
// is set
if (AutoApply) {

  oldStamina = actor.system.stamina.value;
  oldTemp = actor.system.stamina.temporary;
```

```
  dmgType = roll.type;
  ignoredImmunities = [];

  await actor.system.takeDamage(roll.total, { type: dmgType, ignoredImmunities: ignoredImmunities });

  ChatMessage.create({
    speaker: ChatMessage.getSpeaker({ actor }),
    content: `${actor.name}'s stamina changed from ${oldStamina} (${oldTemp} temp) to ${actor.system.stamina.value} (${actor.system.stamina.temporary}
temp).`
  });

}
```

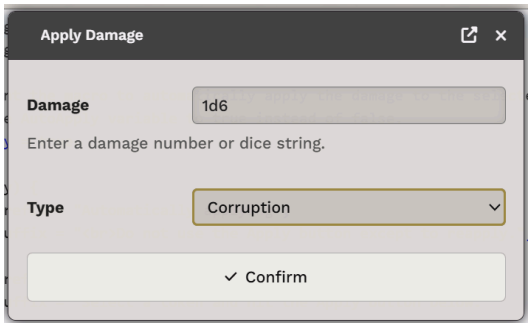
## Apply Damage

This macro can be executed normally, or called with a "damage" parameter.

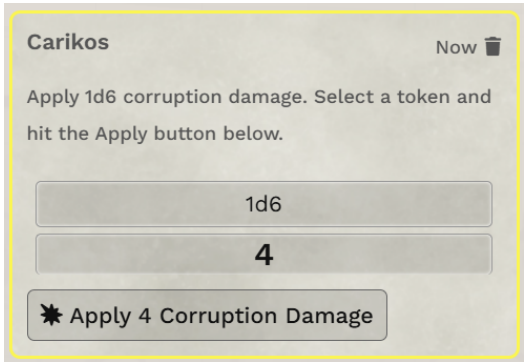
If you call it with a "damage" parameter from the chat window, it will roll that damage. It might be a fixed number, if you want it to always do 5 damage you can use the command "/macro ApplyDamage damage=5". If you want it to do a dice formula like 1d6+1, you would use "/macro ApplyDamage damage=1d6+1". NOTE: In order to be called from chat like this, the macro must be on your hot bar or in the game's macro panel, not a compendium.

If the macro is called without a parameter or executed from the hot bar or Token Action HUD, it will instead popup a dialog window asking for the damage amount, which can be either a specific number or a dice formula. You can also specify the damage type.

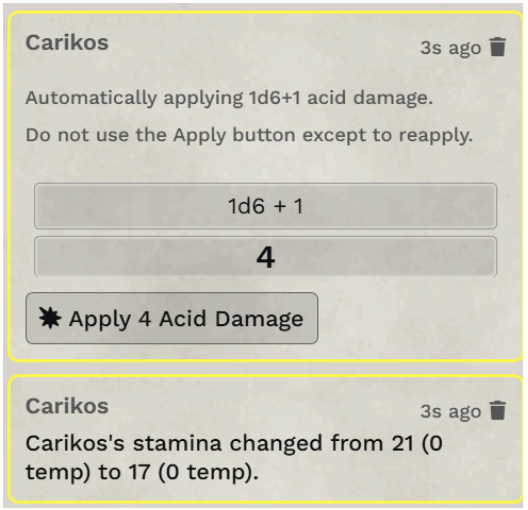
UPDATES: If you want the macro to actually apply the damage to the selected token automatically, without needing to also click the button in chat, set the AutoApply variable to ***true*** at the top of the macro. If you prefer to use the chat button to apply the damage, set the AutoApply variable to ***false***.




**Do Not Auto Apply Damage:**




**Auto Apply Damage:**




2025-11-17: Fixed a bug on the takeDamage() function call that was not correctly applying damage types.

```
// Roll damage and send to the chat window with an Apply Damage button.
```

```
// The dialog box will allow you to specify a damage type.

// If you want the macro to automatically apply the damage to the selected token,
// change the AutoApply variable to true instead of false.
let AutoApply = true;

if (AutoApply) {
  autoApplyPrefix = "Automatically applying";
  autoApplySuffix = "<br>Do not use the Apply button except to reapply.";
} else {
  autoApplyPrefix = "Apply";
  autoApplySuffix = "Select a token and hit the Apply button below.";
}

let roll;

if (!actor) {
  ui.notifications.warn("Please select your token first.");
  return;
}

// Check if this macro is being called with a damage parameter and optional damage type.
// For example, in chat:  /macro ApplyDamage damage=1d6 type=cold
// Valid types are:  acid, cold, corruption, fire, holy, lightning, poison, psychic, sonic
// Note: calling the macro from chat requires this macro to be either in
// quick bar or the game's macro panel, not just a compendium.

if (scope.damage === null || scope.damage === undefined) {

  // No damage parameter was provided so create a popup to collect the roll string.

  const {createFormGroup, createTextInput, createSelectInput} = foundry.applications.fields;
  const content = document.createElement("div");
  const damageForm = createFormGroup({
      label: "Damage",
      hint: "Enter a damage number or dice string.",
      rootId: "damageInput",
      input: createTextInput({ name: "damage" })
  })
  const typeForm = createFormGroup({
    label: "Type",
    rootId: "damageTypeInput",
    input: createSelectInput({
      name: "type",
      type: "checkboxes",
```

```
      options: [
        {label: "Untyped", value: "", selected: true },
        {label: "Acid", value: "acid"},
        {label: "Cold", value: "cold"},
        {label: "Corruption", value: "corruption"},
        {label: "Fire", value: "fire"},
        {label: "Holy", value: "holy"},
        {label: "Lightning", value: "lightning"},
        {label: "Poison", value: "poison"},
        {label: "Psychic", value: "psychic"},
        {label: "Sonic", value: "sonic"}
      ]
    })
  })
  content.append(damageForm, typeForm)

  const fd = await ds.applications.api.DSDialog.input({
      content,
      window: {
          title: "Apply Damage",
          icon: "fa-solid"
      }
  })

  if (!fd) return;

  roll = new ds.rolls.DamageRoll(fd.damage, null, {type: fd.type} );
  await roll.toMessage({
        speaker: ChatMessage.getSpeaker({ token }),
        flavor: `<p>${autoApplyPrefix} ${fd.damage} ${fd.type} damage. ${autoApplySuffix}</p>`,
        rollMode: "roll"
      });

} else {

  // A damage paramter was used, so just roll it.  Use optional damage type if provided.
  let dmgtype = "";

  if (scope.type === null || scope.type === undefined) {
    roll = new ds.rolls.DamageRoll(scope.damage);
  } else {
    roll = new ds.rolls.DamageRoll(scope.damage, null, {type: scope.type});
    dmgtype = scope.type;
  }
```

```
  await roll.toMessage({
      speaker: ChatMessage.getSpeaker({ token: token }),
      flavor: `<p>${autoApplyPrefix} ${fd.damage} ${fd.type} damage. ${autoApplySuffix}</p>`,
      rollMode: "roll"
    });
}

// auto apply damage to token and report the change in stamina if this option
// is set
if (AutoApply) {

  oldStamina = actor.system.stamina.value;
  oldTemp = actor.system.stamina.temporary;
  dmgType = roll.type;
  ignoredImmunities = [];

  await actor.system.takeDamage(roll.total, { type: dmgType, ignoredImmunities: ignoredImmunities });

  ChatMessage.create({
    speaker: ChatMessage.getSpeaker({ actor }),
    content: `${actor.name}'s stamina changed from ${oldStamina} (${oldTemp} temp) to ${actor.system.stamina.value} (${actor.system.stamina.temporary}
temp).`
  });

}
```

## Apply 1d10 Unresisted Psychic Damage

By request in the discord, here's a macro that does 1d10 irresistible psychic damage.  If it is run by a player who is assigned a character they own, it will auto-apply the damage to that owned character regardless of whether any token is selected.  If it is run by someone who doesn't have an assigned character, it still just sends the damage to chat and you will need to apply it later by selecting a token then clicking the chat button.

If you want to change the amount or the type, edit the variables at the top of the macro.  If you do not want it to ignore immunity, delete the "psychic" from ignoredImmunities.

UPDATES:

2025-11-17:  Fixed a bug on the takeDamage() function call that was not correctly applying damage types.

```
// Roll damage and send to the chat window with an Apply Damage button.
// if the player has an assigned character, it will automatically apply the
// rolled damage to that character and report the new stamina.

let myPC = game.user.character;
```

```
let type = "psychic";
let amount = '1d10';
let ignoredImmunities = ["psychic"];

let roll = new ds.rolls.DamageRoll(amount, null, {type: type});

if (myPC === null) {
  await roll.toMessage({
      speaker: game.user,
      flavor: `<p>${game.user.name} does not own a token.  Rolling ${amount} ${type} damage, use the Apply button on a selected token.</p>`,
      rollMode: "roll"
    });
} else {
  oldStamina = myPC.system.stamina.value;
  oldTemp = myPC.system.stamina.temporary;

  await roll.toMessage({
      speaker: ChatMessage.implementation.getSpeaker({ actor: character }),
      flavor: `<p>Automatically applying ${amount} ${type} damage to ${myPC.name}. Do not use the Apply button unless you need to reapply it again.</p>`,
      rollMode: "roll"
    });

  await myPC.system.takeDamage(roll.total, { type: type, ignoredImmunities: ignoredImmunities });

  ChatMessage.create({
    speaker: ChatMessage.getSpeaker({ actor }),
    content: `${actor.name}'s stamina changed from ${oldStamina} (${oldTemp}) to ${myPC.system.stamina.value} (${myPC.system.stamina.temporary}).`
  });

}
```

## Automate Bloodbound Bands Damage

Ady has a macro to automatically apply the 1 point damage to each other bound PC wearing a bloodbound band.

## Quick Strike Damage Macros

Stevil has two versions of the Apply Damage macros for his Quick Strike module, which applies the damage to the *targeted* token instead of the *selected* token.

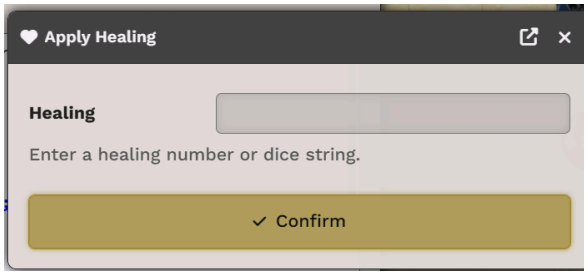
# Healing Macros

## ApplyHealing

This macro can be executed normally, or called with a "healing" parameter.

If you call it with a "healing" parameter from the chat window, it will roll that healing amount.  It might a fixed number, if you want it to always do 5 healingyou can use the command "/macro ApplyHealing healing=5".  If you want it to do a dice formula like 1d6+1, you would use "/macro ApplyHealing healing=1d6+1".  NOTE:  In order to be called from chat like this, the macro must be on your hot bar or in the game's macro panel, not a compendium.

If the macro is called without a parameter or executed from the hot bar or Token Action HUD, it will instead popup a dialog window asking for the healing amount, which can be either a specific number or a dice formula.




```
// Roll healing (or assign a specific value) and send to the chat window with a
// Regain Stamina button.

let roll;

if (!actor) {
  ui.notifications.warn("Please select your token first.");
  return;
}

// Check if this macro is being called with a healing parameter.
// For example, in chat:  /macro ApplyHealing healing=1d6
// Note: calling the macro from chat requires this macro to be either in
// quick bar or the game's macro panel, not just a compendium.

if (scope.healing === null || scope.healing === undefined) {

  // No healing parameter was provided so create a popup to collect the roll string.

  const {createFormGroup, createTextInput} = foundry.applications.fields;
  const content = document.createElement("div");
  const healingForm = createFormGroup({
      label: "Healing",
      hint: "Enter a healing number or dice string.",
```

```
    rootId: "healingInput",
    input: createTextInput({ name: "healing" })
  })
  content.append(healingForm)

  const fd = await ds.applications.api.DSDialog.input({
    content,
    window: {
      title: "Apply Healing",
      icon: "fa-solid fa-heart"
    }
  })

  if (!fd) return;

  roll = new ds.rolls.DamageRoll(fd.healing, null, {type: "value", isHeal: true } );
  roll.toMessage({
      author: game.user,
      speaker: ChatMessage.getSpeaker({ token }),
      flavor: `<p>Apply Healing: ${fd.healing}</p>`,
      rollMode: "roll"
    });

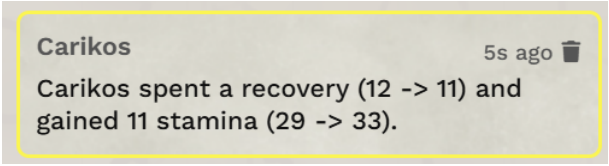
} else {

  // A healing paramter was used, so just roll it.
  roll = new ds.rolls.DamageRoll(scope.healing, null, {type: "value", isHeal: true } );

  await roll.toMessage({
    user: game.user._id,
    speaker: ChatMessage.getSpeaker({ token: token }),
    flavor: `<p>Apply Healing: ${healing}</p>`,
    rollMode: "roll"
   });
}
```

## Spend Recovery

This macro is based on one from KJTailor and finaljas90 in the discord. It will spend one of the selected token's recoveries and increase their stamina by the recovery value, capped at their maximum stamina.




```
// This macro spends one recovery and adds your recovery value to your stamina,
// limited by your min and max stamina.

if (!actor) return ui.notifications.error("No token selected");
if (actor.type !== "hero") return ui.notifications.error("Only heroes can use recoveries");

const recoveries = actor.system.recoveries;

if (recoveries.value === 0) return ui.notifications.error("The actor has no recoveries to use");

const stamina = actor.system.stamina;

const oldRecoveries = recoveries.value;
const oldStamina = stamina.value;
const newRecoveries = recoveries.value - 1;
const newStamina = Math.clamp(stamina.value + recoveries.recoveryValue, stamina.min ?? -Infinity,
stamina.max);

await actor.update({
  "system.stamina.value": newStamina,
  "system.recoveries.value": newRecoveries,
});

ChatMessage.create({
  author: game.user,
  speaker: ChatMessage.getSpeaker({ actor }),
  content: `${actor.name} spent a recovery (${oldRecoveries} -> ${newRecoveries}) and gained
${recoveries.recoveryValue} stamina (${oldStamina} -> ${newStamina}).`
});
```

## Alternate Spend Recovery

Spend Recovery (including Bloodbound Bands) by Ady

# My Life For Yours / Spend Your Recovery For Allies

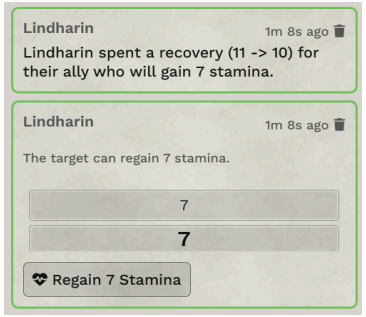
This macro can be used by Censors for the "My Life For Yours" ability, or other similar effects. The selected character spends one of their recoveries and grants their recovery value as free healing to a target.

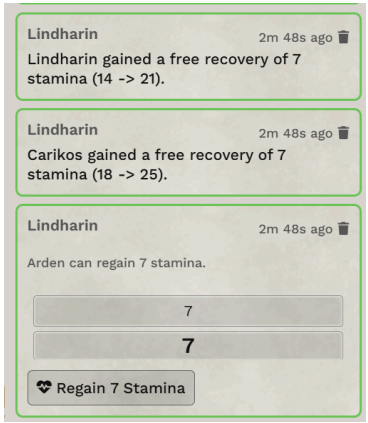
By default, it will spend one recovery from the selected token and then send a chat message with a healing button which the recipient can click to heal their own character, as shown in the first picture.

If the user has an ally's token targeted (using the T key by default), then it will identify the target in the heal chat card.

If the user has sufficient ownership to the targeted ally (which can be themself), then it will automatically apply the healing to that ally rather than displaying a button that needs to be clicked.

There is a configuration setting in the macro that lets you select multiple targets. If that is turned on, and you have multiple targets, it will auto-apply the healing to any targets you own, then list a chat card with a button for any targets you do not own. It will name the tokens in either case, as shown in the second picture. The user had owner permissions for Lindharin and Carikos, but not for Arden.




```
// This macro is for the Censor ability My Life For Yours, or similar abilities
// where a hero can spend one of their own recoveries to grant a free recovery
// using the original hero's recovery value.
//
// If the hero's player has owner access to the target, you can use the built-in
// Foundry targeting feature to target the designated ally and the recovery will be
// fully automated.  If the hero's player does not own the target, or there is no
// designated target it will instead display a chat card with an "Apply Healing" button
// and the owner of the ally token will need to select their own token and use the button.

//-------------------------------------------------------------------------------
// By default this macro will only process the healing for one designated target.
// If you need to be able to target multiple tokens and heal them all, change the
// next line to true instead of false.
const allowMultipleTargets = false;
```

```
//-------------------------------------------------------------------------------
// Spend a recovery from the active selected token.
if (!actor) return ui.notifications.error("No token selected");
if (actor.type !== "hero") return ui.notifications.error("Only heroes can use recoveries");

const recoveries = actor.system.recoveries;

if (recoveries.value === 0) return ui.notifications.error("The actor has no recoveries to use");

const oldRecoveries = recoveries.value;
const newRecoveries = recoveries.value - 1;
const recoveryValue = recoveries.recoveryValue;

await actor.update({
  "system.recoveries.value": newRecoveries,
});

ChatMessage.create({
  author: game.user,
  speaker: ChatMessage.getSpeaker({ actor }),
  content: `${actor.name} spent a recovery (${oldRecoveries} -> ${newRecoveries}) for their ally who will gain
${recoveryValue} stamina.`
});

//-------------------------------------------------------------------------------
// Handle the healing for the targets.

const targets = (allowMultipleTargets) ? game.user.targets : [game.user.targets.first()];
let roll;
let unprocessed = "";

if (targets.size == 0) {
  // There are no targets, we'll just create a generic "Apply Healing" button at the end.
  unprocessed = "The target";

} else {
  // We do have targets.  Loop through any hero targets and see if we're the owner
  // of any of them. If we are the owner, then just apply the recovery value directly.
  // If we are not the owner, add them to the list of unprocessed targets and we'll
  // create a roll chat message at the end.

  targets
    .filter( (target) => target.actor.type === "hero" )
    .forEach(async (target) => {
```

```
      if (!target.isOwner) {
        // the user is not the owner so just add them to the list of unprocessed targets
        unprocessed += (unprocessed === "") ? target.name : " and " + target.name;

      } else {
        // the user is the owner of the target, so just heal them and send a log to chat
        oldStamina = target.actor.system.stamina.value;
        newStamina = Math.clamp(target.actor.system.stamina.value + recoveries.recoveryValue,
target.actor.system.stamina.min ?? -Infinity, target.actor.system.stamina.max);

        ChatMessage.create({
          author: game.user,
          speaker: ChatMessage.getSpeaker({ actor }),
          content: `${target.actor.name} gained a free recovery of ${recoveryValue} stamina (${oldStamina} ->
${newStamina}).`
        });

        await target.actor.update({
          "system.stamina.value": newStamina
        });

      }
    })

}

//-----------------------------------------------------------------------------
// if there are any unprocessed targets, create a healing roll chat card so they can
// manually apply the healing.

if ( unprocessed !== "" ) {
  // there are targets not owned by the actor so create a chat card
  // with a button to apply the healing.

  roll = new ds.rolls.DamageRoll(recoveryValue.toString(), null, {type: "value", isHeal: true } );

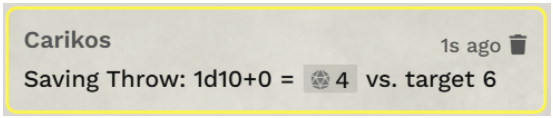
  roll.toMessage({
        author: game.user,
        speaker: ChatMessage.getSpeaker({ token }),
        flavor: `<p>${unprocessed} can regain ${recoveryValue} stamina.</p>`,
        rollMode: "roll"
      });

}
```

# Saving Throws

## Roll A Saving Throw

This macro rolls a saving throw.  If one or more tokens are selected, it will roll for each one individually using any configured bonuses or altered target numbers.  You will need to end the effect manually; the macro only makes the roll.




```
// Roll a saving throw.  If one or more tokens are selected, it will roll once per
// token and use any configured saving throw bonuses or altered target numbers.

let saveBonus;
let threshhold;

if (!actor) {
  saveBonus = 0;
  threshhold = 6;
  ChatMessage.create({
    author: game.user,
    speaker: ChatMessage.getSpeaker({ actor }),
    content: `Saving Throw: 1d10+${saveBonus} = [[1d10+${saveBonus}]] vs. target ${threshhold}`
  });
} else {

  const actors = canvas.tokens.controlled.map((t) => t.actor); // update all selected tokens

  actors
    .forEach(async (actor) => {
      saveBonus = actor.system.combat.save.bonus;
      threshhold = actor.system.combat.save.threshold;
      ChatMessage.create({
        author: game.user,
        speaker: ChatMessage.getSpeaker({ actor }),
        content: `Saving Throw: 1d10+${saveBonus} = [[1d10+${saveBonus}]] vs. target ${threshhold}`
      });
  });
}
```

## Roll All "Save Ends" Saving Throws

This macro is based on code from @ChaosOS in the discord. It will process all selected tokens to find any active effects with a duration of "save ends". For each effect it will pop up a dialog box asking for any save modifiers or altered success thresholds. The actor name and effect name will be listed in the dialog window's title. You can close any dialog that you do not want to roll a save for, or submit the dialog to have it make the roll and end the effect if successful.

```
// This macro will make a saving throw for each active effect with a "save ends" duration
// for each selected token. Each effect will have a popup dialog to ask for any special
// modifiers.

if (!actor) {
  ui.notifications.warn("Please select your token first.");
  return;
}

let effects;
const actors = canvas.tokens.controlled.map((t) => t.actor); // update all selected tokens

actors
  .forEach(async (actor) => {
    // get all active effects with a save ends duration
    effects = actor.effects.filter(e => e.system.end.type === "save" && e.active);
    if (!effects.length) return ui.notifications.warn("No active save ends effects found")

    // now you have an array of effects. For each one, pop up a dialog to allow
    // a save.  You can override the roll and/or success threshold as needed.
    // The effect name is in the window title in case the modifiers vary by
    // effect.  If you want to skip one, just close out of that dialog window.
    effects.forEach(e => e.system.rollSave({}, {window: {title:`${actor.name} : ${e.name}`} } ))
  });
```

# Combat Effects

## Aid Attack

Aid Attack by Ady

## Death Tracker

This Death Tracker macro by Ady clears NPC tokens from the field when they have the Dead (applied manually for minions) or Dying (applied automatically at 0 or less HP). Removes them from the combat tracker too, and leaves a skull behind which can be cleared at the end of combat with a toggle.

### POWER WORD: KILL Macro

**(Director Tool for the Death Tracker)**

This supplemental macro for the death tracker lets you mass kill NPCs, applies the Dead condition to all targeted/selected creatures, prioritizing targeted. Doesn't affect players.

## Grab

Grab macro by Ady, includes ability to move with an NPC, handle escapes, etc.

## I'm No Threat

Harlequin's I'm No Threat by Ady.  It automates the I'm No Threat ability's edge and disengage buffs, and gives a cool visual by actually swapping your token art.

## Marked & Judgement Macros

The Marked & Judgement macros by Ady apply the appropriate condition to the targeted creature and then have a reminder that holds until you end combat or refresh the tab which reminds you to reapply your Mark/Judgement when the creature they were on gains the Dead condition.

## Triggered Action Tracker

Ady has posted a Triggered Action Tracker macro.  If there's an active encounter, applies a condition which gets automatically cleared whenever a triggered action is posted, reapplies at round start.

## Apply Effect Macro

ChaosOS posted a macro to apply an effect.

# Map Management

## Scene Grid Conversion

ChaosOS posted a macro to convert scenes with 5' grids to use the Draw Steel scale that just uses "squares" instead of feet.  It updates the scene as well as any lights, sounds, regions, etc.

# Token Management

## Add Basic Abilities to Actors

Ady has a macro to add almost all basic abilities to actors.  Intended for NPCs, it excludes the heal, catch a breath, and free strikes since those do not apply to monsters.

## Add Token to Combat Initiative Group and Apply Token Ring Color

There is a macro from Izzy in the Foundry Draw Steel discord channel that adds selected tokens to a combat initiative group and also applies a token ring color.

## Apply Token Ring Color from Color Picker

There is a macro from Zhell in the Foundry Draw Steel discord that will prompt you with a color picker to select a color that it applies the token ring color of all selected tokens.

## Apply Token Ring Color from Dropdown

This is a variation on the prior macro from Zhell that uses a pre-defined drop-down of colors instead of a color picker. The specific options and colors were based on a request from KJTailor in the discord channel for this macro collection.  You can edit the rows inside the colorSelect options to add or remove entries, rename the labels of existing entries, or alter the color hex codes to your own color scheme.  The labels are what will show in the drop-down, while the values are the color hex codes that get applied in the tokens' ring settings.

```
const tokens = canvas.tokens.controlled.map(token => token.document);
if (tokens.length < 1) return void ui.notifications.error("Must have creatures selected.");

const colorSelect = foundry.applications.fields.createSelectInput({
      name: "color",
      type: "checkboxes",
      options: [
        {label: "Hero", value: "#0011ff"},
        {label: "Civilian", value: "#737373"},
        {label: "Ally", value: "#04ff00"},
        {label: "Minion", value: "#ffc800"},
        {label: "Horde/Platoon/Elite", value: "#ff7b00"},
        {label: "Leader", value: "#ff0000"},
        {label: "Mount", value: "#ff0095"},
        {label: "Solo", value: "#9900ff"}
      ]
});
const colorForm = foundry.applications.fields.createFormGroup({ input: colorSelect, label: "Color" });
const data = await ds.applications.api.DSDialog.input({
  window: { title: `Apply Token Ring Color` },
  content: colorForm.outerHTML,
});
```

```
if (!data) return;
const updates = tokens.map(token => {
  return { _id: token.id, "ring.colors.ring": Color.fromString(data.color), "ring.enabled": true };
});
await TokenDocument.implementation.updateDocuments(updates, { parent: canvas.scene });
```

### Apply Visual Icons to Token Groups

[Apply Visual Icons/Effects to Identify Tokens in Combat Group] by Ady

## Downtime Activities

### Fishing

ColinGreenleaf created a [fishing macro] in the discord.

### Creating Foundry Treasure Items for Imbued Armor/Implements/Weapons

Cora created macros that with automatically create a Foundry treasure item using the rules for imbuing Armor, Implements and Weapons.

[MCDM DS-Creations dicord link]

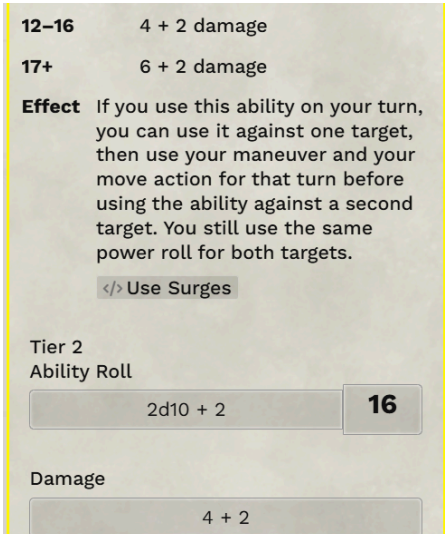
## Adding Macro Buttons to Abilities

You can add a clickable button for macros to the Effect text of Draw Steel abilities.  While editing an ability, drag-and-drop a macro into the Effect field. When the ability is used, there will be a button for the macro included in the chat card.

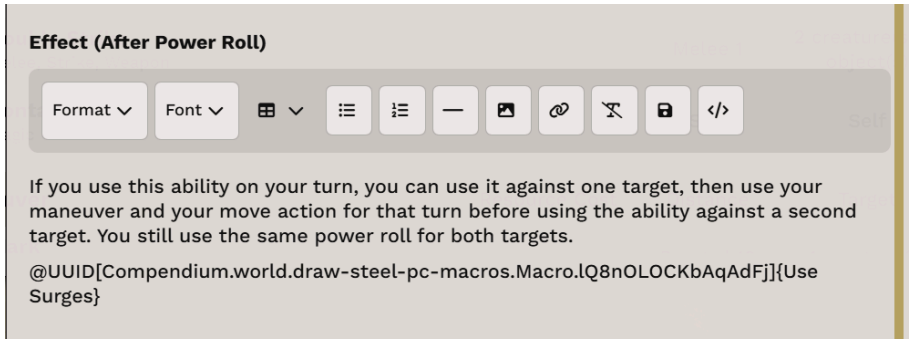
When you drag the macro into the Effect field, it will be converted to a string like this:

@UUID[Compendium.world.draw-steel-pc-macros.Macro.lQ8nOLOCKbAqAdFj]{Use Surges}

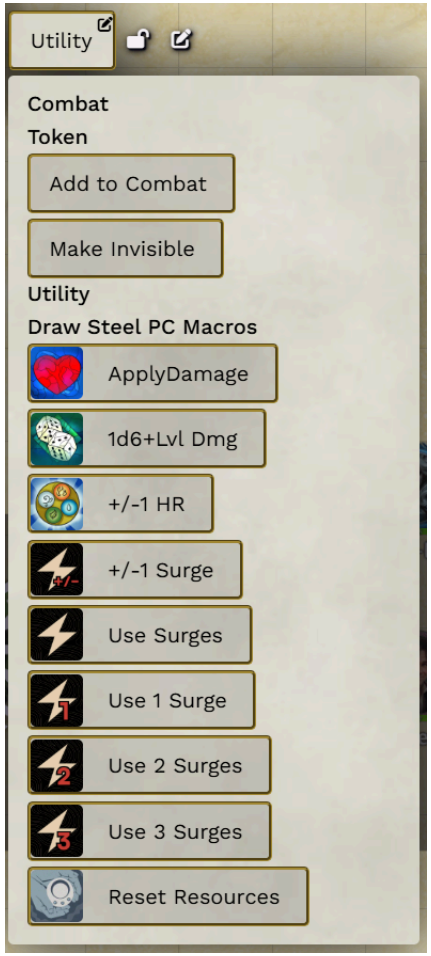
The section at the end inside the curly braces will be used as the text for the button when the macro is displayed, so you can alter the way the button appears.



For example, the output will look like the "Use Surges" button at the end of the Effect in the screenshot to the left.



# Adding Macros to Token Action HUD




If you are using the [Token Action HUD Draw Steel module] from Dragodoth, you can add macros to the token action HUD.

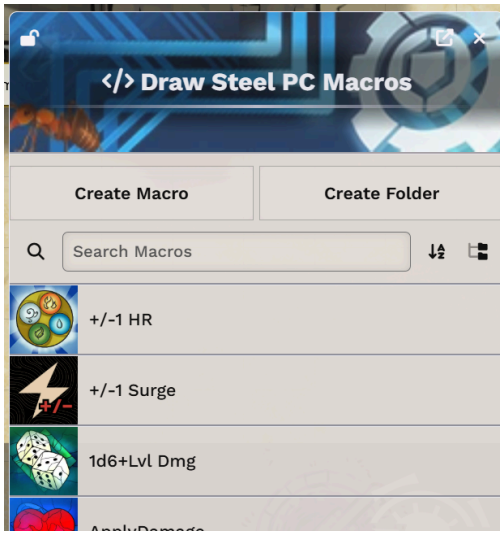
In my examples, I created a macro compendium called Draw Steel PC Macros, and put all of the above macros in it. Then I edited the Utility button on the HUD and added that "Group: Draw Steel PC Macros" group. I then re-ordered the macros inside the "Draw Steel PC Macros" sub-section of the Utility menu to have the macros in the order I preferred. Follow these steps and see the screenshots for what it looks like.

1. In the Foundry Configuration screen go to the Token Action HUD Core section and make sure the settings "Enable Compendium Actions" and "Enable Macro Actions" are turned on.
2. Create a compendium for macros. In my example, I created a compendium called Draw Steel PC Macros. You can have more than one compendium if necessary to split up macros among multiple sections of the HUD.

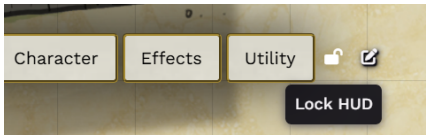
   If your players have trouble using macros in the compendium, you might check the permissions by right clicking on the compendium and using the Configure Ownership option. I think they only need Observer access, but I'm not 100% sure.

   NOTE: It's been reported that you do not need to use a compendium, but can create a folder in the Macro panel in the game world and set it to be Observer access for players, and that will work in TAH too.

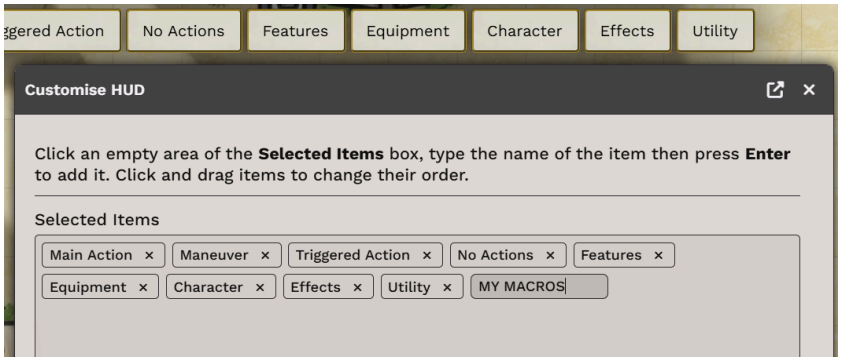
3. Import or create your macros in that compendium.



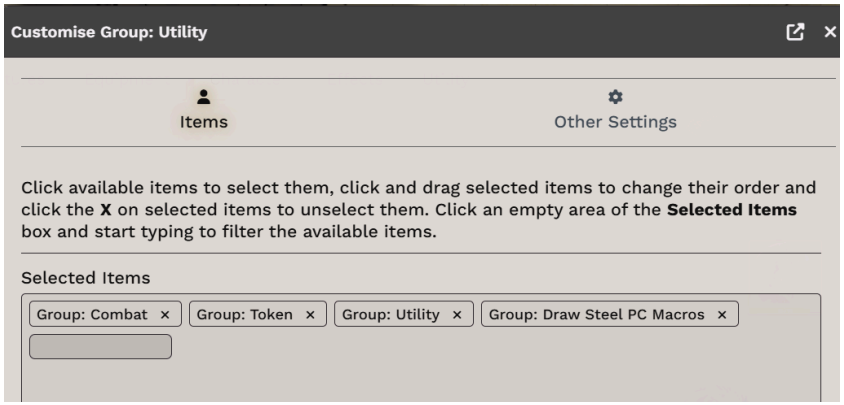
4. Unlock the Token Action HUD bar by hovering your mouse over it until you see the lock icon just to the right of the last HUD button. If the lock icon is a closed lock, click it to unlock it.



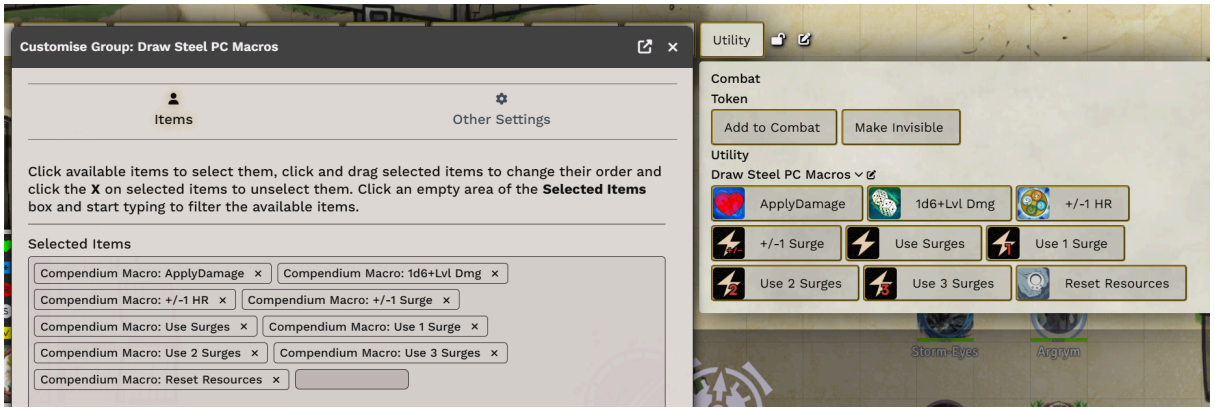

5. Decide which HUD button you want the macros to appear under. If you want the macros to show up under an existing button skip this step.
If you want to create a new HUD button for macros, click the pencil icon next to the lock icon. In the Customize HUD popup window, in the Selected Items section of the panel, click in the gray rectangle at the end of the list of button names, and type your new button name. For example, you could call the new button "My Macros", like in the picture below. Once the button name is entered, you can drag-and-drop the buttons to put them in whatever order you want. Click Save and Close.




6. Right click on whichever button you want to have the macros. In the Customize Group popup window, click in the blank gray rectangle at the end of the Selected Items panel. Start typing in the name of your macro compendium. As you type, it should filter down the list of all available groups in the "Available Items" panel. Once you find your compendium macros group in the Available Items panel, click on it and it will add it to the Selected Items panel. You can then drag-and-drop to sort it with any other groups in the same HUD button. Click Save and Close.




7. If you want to re-order the macros inside the HUD drop-down menu, you can do that too. First, hover your mouse over the HUD button but do not right-click on that button. Move the mouse down to hover over the sub-heading that matches your compendium macros name, and either click the tiny pencil icon next to the subheading or just right click on that sub-heading. It will open the Customize Group panel for your compendium macros. You can then re-order the individual macros or delete specific ones that aren't relevant for your character. Click Save and Close.




8. When you're done modifying the HUD, remember to lock it again to prevent accidental edits. Hover your mouse over the HUD until you see the lock icon, and click it to lock.
9. This process can be done individually by each player. There is an option to set a global default layout in the settings for Token Action HUD Core, but I haven't used it yet so I don't have instructions for that yet.

## Macro Design Notes

The list of pre-defined top level variables were identified by ChaosOS in the discord:
"The top level variables available to you already are:

- **actor** & **token** (Defaults to "first" selected token, falls back to game.user.character)
- **speaker** (equal to ChatMessage.getSpeaker({ actor, token })

- **character** (equal to game.user.character)

Those are just always provided to any script macro as pre-assigned variables.

Some properties, like actor.system.movement.types, are sets which get stripped by JSON.stringify and show in the console as empty.  If you want to get the "raw data" you should be using actor.toObject().

There's guidance on how to create a macro to:

- spawn an actor
- manage a Summoner's minions
- update combatant groups.
- apply an effect (like Marked) to a target or have a pop-up that creates enrichers for conditions

Auto Gain HR at EoT - World Module

ChaosOS posted a code snippet that can be used in a world module to trigger at end of turn:

```
Hooks.on("combatTurn", (combat, updateData, updateOptions) => {
  const actor = combat.combatant?.actor;
  if (actor?.type !== "hero") return;
  actor.modifyTokenAttribute("hero.primary.value", 3, true);
});
```

ChaosOS posted a code snippet that can alter how potencies are calculated:

```
Hooks.on("ds.prepareActorData", (actor) => {
  const potency = foundry.utils.getProperty(actor, "system.potency")
  if (potency) {
    potency.weak = 0;
    potency.average = Math.floor(potency.strong / 2);
  }
});
```