Human-in-the-loop Tool Validation

This walkthrough demonstrates how to add Human validation to any Tool. We'll do this using the `HumanApprovalCallbackhandler`.

Let's suppose we need to make use of the ShellTool. Adding this tool to an automated flow poses obvious risks. Let's see how we could enforce manual human approval of inputs going into this tool.

Note: We generally recommend against using the ShellTool. There's a lot of ways to misuse it, and it's not required for most use cases. We employ it here only for demonstration purposes.

```python
from langchain.callbacks import HumanApprovalCallbackHandler
from langchain.tools import ShellTool
```

API Reference:

- HumanApprovalCallbackHandler from `langchain.callbacks`
- ShellTool from `langchain.tools`

```python
tool = ShellTool()
```

```python
print(tool.run("echo Hello World!"))
```

```
    Hello World!
```

Adding Human Approval

Adding the default HumanApprovalCallbackHandler to the tool will make it so that a user has to manually approve every input to the tool before the command is actually executed.

```
tool = ShellTool(callbacks=[HumanApprovalCallbackHandler()])
```

```
print(tool.run("ls /usr"))
```

```
    Do you approve of the following input? Anything except 'Y'/'Yes'
(case-insensitive) will be treated as a no.

    ls /usr
    yes
    X11
    X11R6
    bin
    lib
    libexec
    local
    sbin
    share
    standalone
```

```
print(tool.run("ls /private"))
```

```
    Do you approve of the following input? Anything except 'Y'/'Yes'
(case-insensitive) will be treated as a no.

    ls /private
    no



    ----------------------------------------------------------------------
-----

    HumanRejectedException                    Traceback (most recent call
last)

    Cell In[17], line 1
    ----> 1 print(tool.run("ls /private"))
    File ~/langchain/langchain/tools/base.py:257, in BaseTool.run(self,
tool_input, verbose, start_color, color, callbacks, **kwargs)
        255 # TODO: maybe also pass through run_manager is _run supports
kwargs
        256 new_arg_supported =
signature(self._run).parameters.get("run_manager")
    --> 257 run_manager = callback_manager.on_tool_start(
        258     {"name": self.name, "description": self.description},
        259     tool_input if isinstance(tool_input, str) else
str(tool_input),
        260     color=start_color,
        261     **kwargs,
        262 )
        263 try:
        264     tool_args, tool_kwargs =
self._to_args_and_kwargs(parsed_input)


    File ~/langchain/langchain/callbacks/manager.py:672, in
CallbackManager.on_tool_start(self, serialized, input_str, run_id,
parent_run_id, **kwargs)
        669 if run_id is None:
        670     run_id = uuid4()
    --> 672 _handle_event(
        673     self.handlers,
        674     "on_tool_start",
        675     "ignore_agent",
        676     serialized,
        677     input_str,
        678     run_id=run_id,
        679     parent_run_id=self.parent_run_id,
        680     **kwargs,
        681 )
        683 return CallbackManagerForToolRun(
        684     run_id, self.handlers, self.inheritable_handlers,
self.parent_run_id
        685 )


    File ~/langchain/langchain/callbacks/manager.py:157, in
_handle_event(handlers, event_name, ignore_condition_name, *args,
**kwargs)
```

```
        155 except Exception as e:
        156     if handler.raise_error:
    --> 157         raise e
        158     logging.warning(f"Error in {event_name} callback: {e}")


    File ~/langchain/langchain/callbacks/manager.py:139, in
_handle_event(handlers, event_name, ignore_condition_name, *args,
**kwargs)
        135 try:
        136     if ignore_condition_name is None or not getattr(
        137         handler, ignore_condition_name
        138     ):
    --> 139         getattr(handler, event_name)(*args, **kwargs)
        140 except NotImplementedError as e:
        141     if event_name == "on_chat_model_start":


    File ~/langchain/langchain/callbacks/human.py:48, in
HumanApprovalCallbackHandler.on_tool_start(self, serialized, input_str,
run_id, parent_run_id, **kwargs)
         38 def on_tool_start(
         39     self,
         40     serialized: Dict[str, Any],
       (...)
         45     **kwargs: Any,
         46 ) -> Any:
         47     if self._should_check(serialized) and not
self._approve(input_str):
    ---> 48         raise HumanRejectedException(
         49             f"Inputs {input_str} to tool {serialized} were
rejected."
         50         )


    HumanRejectedException: Inputs ls /private to tool {'name':
'terminal', 'description': 'Run shell commands on this MacOS machine.'}
were rejected.
```

Configuring Human Approval

Let's suppose we have an agent that takes in multiple tools, and we want it to only trigger human approval requests on certain tools and certain inputs. We can configure out callback handler to do just this.

```
from langchain.agents import load_tools
from langchain.agents import initialize_agent
from langchain.agents import AgentType
from langchain.llms import OpenAI
```

API Reference:

- load_tools from `langchain.agents`
- initialize_agent from `langchain.agents`
- AgentType from `langchain.agents`
- OpenAI from `langchain.llms`

```
def _should_check(serialized_obj: dict) -> bool:
    # Only require approval on ShellTool.
    return serialized_obj.get("name") == "terminal"


def _approve(_input: str) -> bool:
    if _input == "echo 'Hello World'":
        return True
    msg = (
        "Do you approve of the following input? "
        "Anything except 'Y'/'Yes' (case-insensitive) will be treated as a
no."
    )
    msg += "\n\n" + _input + "\n"
    resp = input(msg)
    return resp.lower() in ("yes", "y")


callbacks = [HumanApprovalCallbackHandler(should_check=_should_check,
approve=_approve)]
```

```
llm = OpenAI(temperature=0)
tools = load_tools(["wikipedia", "llm-math", "terminal"], llm=llm)
```

```
agent = initialize_agent(
    tools,
    llm,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
)
```

```
agent.run(
    "It's 2023 now. How many years ago did Konrad Adenauer become
Chancellor of Germany.",
    callbacks=callbacks,
)
```

```
    'Konrad Adenauer became Chancellor of Germany in 1949, 74 years ago.'
```

```
agent.run("print 'Hello World' in the terminal", callbacks=callbacks)
```

```
    'Hello World'
```

```
agent.run("list all directories in /private", callbacks=callbacks)
```

```
    Do you approve of the following input? Anything except 'Y'/'Yes'
(case-insensitive) will be treated as a no.

    ls /private
    no




    ---------------------------------------------------------------------
-----

    HumanRejectedException                    Traceback (most recent call
last)

    Cell In[39], line 1
    ----> 1 agent.run("list all directories in /private",
callbacks=callbacks)


    File ~/langchain/langchain/chains/base.py:236, in Chain.run(self,
callbacks, *args, **kwargs)
        234     if len(args) != 1:
        235         raise ValueError("`run` supports only one positional
argument.")
    --> 236     return self(args[0], callbacks=callbacks)
[self.output_keys[0]]
        238 if kwargs and not args:
        239     return self(kwargs, callbacks=callbacks)
[self.output_keys[0]]


    File ~/langchain/langchain/chains/base.py:140, in Chain.__call__(self,
inputs, return_only_outputs, callbacks)
        138 except (KeyboardInterrupt, Exception) as e:
        139     run_manager.on_chain_error(e)
    --> 140     raise e
        141 run_manager.on_chain_end(outputs)
        142 return self.prep_outputs(inputs, outputs, return_only_outputs)


    File ~/langchain/langchain/chains/base.py:134, in Chain.__call__(self,
inputs, return_only_outputs, callbacks)
        128 run_manager = callback_manager.on_chain_start(
        129     {"name": self.__class__.__name__},
        130     inputs,
        131 )
        132 try:
        133     outputs = (
    --> 134         self._call(inputs, run_manager=run_manager)
        135         if new_arg_supported
        136         else self._call(inputs)
        137     )
        138 except (KeyboardInterrupt, Exception) as e:
        139     run_manager.on_chain_error(e)


    File ~/langchain/langchain/agents/agent.py:953, in
AgentExecutor._call(self, inputs, run_manager)
        951 # We now enter the agent loop (until it returns something).
        952 while self._should_continue(iterations, time_elapsed):
```

```
    --> 953     next_step_output = self._take_next_step(
        954         name_to_tool_map,
        955         color_mapping,
        956         inputs,
        957         intermediate_steps,
        958         run_manager=run_manager,
        959     )
        960     if isinstance(next_step_output, AgentFinish):
        961         return self._return(
        962             next_step_output, intermediate_steps,
run_manager=run_manager
        963         )


    File ~/langchain/langchain/agents/agent.py:820, in
AgentExecutor._take_next_step(self, name_to_tool_map, color_mapping,
inputs, intermediate_steps, run_manager)
        818         tool_run_kwargs["llm_prefix"] = ""
        819     # We then call the tool on the tool input to get an
observation
    --> 820     observation = tool.run(
        821         agent_action.tool_input,
        822         verbose=self.verbose,
        823         color=color,
        824         callbacks=run_manager.get_child() if run_manager else
None,
        825         **tool_run_kwargs,
        826     )
        827 else:
        828     tool_run_kwargs = self.agent.tool_run_logging_kwargs()


    File ~/langchain/langchain/tools/base.py:257, in BaseTool.run(self,
tool_input, verbose, start_color, color, callbacks, **kwargs)
        255 # TODO: maybe also pass through run_manager is _run supports
kwargs
        256 new_arg_supported =
signature(self._run).parameters.get("run_manager")
    --> 257 run_manager = callback_manager.on_tool_start(
        258     {"name": self.name, "description": self.description},
        259     tool_input if isinstance(tool_input, str) else
str(tool_input),
        260     color=start_color,
        261     **kwargs,
        262 )
        263 try:
        264     tool_args, tool_kwargs =
self._to_args_and_kwargs(parsed_input)


    File ~/langchain/langchain/callbacks/manager.py:672, in
CallbackManager.on_tool_start(self, serialized, input_str, run_id,
parent_run_id, **kwargs)
        669 if run_id is None:
        670     run_id = uuid4()
    --> 672 _handle_event(
        673     self.handlers,
        674     "on_tool_start",
        675     "ignore_agent",
        676     serialized,
        677     input_str,
        678     run_id=run_id,
        679     parent_run_id=self.parent_run_id,
        680     **kwargs,
        681 )
        683 return CallbackManagerForToolRun(
        684     run_id, self.handlers, self.inheritable_handlers,
self.parent_run_id
        685 )


    File ~/langchain/langchain/callbacks/manager.py:157, in
_handle_event(handlers, event_name, ignore_condition_name, *args,
**kwargs)
        155 except Exception as e:
        156     if handler.raise_error:
    --> 157         raise e
        158     logging.warning(f"Error in {event_name} callback: {e}")


    File ~/langchain/langchain/callbacks/manager.py:139, in
_handle_event(handlers, event_name, ignore_condition_name, *args,
**kwargs)
        135 try:
        136     if ignore_condition_name is None or not getattr(
        137         handler, ignore_condition_name
        138     ):
    --> 139         getattr(handler, event_name)(*args, **kwargs)
```

```
        140 except NotImplementedError as e:
        141     if event_name == "on_chat_model_start":


    File ~/langchain/langchain/callbacks/human.py:48, in
HumanApprovalCallbackHandler.on_tool_start(self, serialized, input_str,
run_id, parent_run_id, **kwargs)
         38 def on_tool_start(
         39     self,
         40     serialized: Dict[str, Any],
       (...)
         45     **kwargs: Any,
         46 ) -> Any:
         47     if self._should_check(serialized) and not
self._approve(input_str):
    ---> 48         raise HumanRejectedException(
         49             f"Inputs {input_str} to tool {serialized} were
rejected."
         50         )


    HumanRejectedException: Inputs ls /private to tool {'name':
'terminal', 'description': 'Run shell commands on this MacOS machine.'}
were rejected.
```