# Jira Issue Details (PF Tools Screen Compatible)

# Pattern expected:

# def your\_function\_name(your\_arguments):

# ...

# if \_\_name\_\_ == "\_\_main\_\_":

# result = your\_function\_name(your\_arguments)

# print(result)

import requests

import json

import base64

from typing import Dict, Any, List, Optional

# Set these for your Jira Cloud site and credentials

JIRA\_BASE\_URL = 'https://your-domain.atlassian.net' # e.g., https://example.atlassian.net

JIRA\_USER\_EMAIL = 'user@example.com' # Atlassian account email

JIRA\_API\_TOKEN = 'your\_jira\_api\_token\_here' # Create at https://id.atlassian.com/manage-profile/security/api-tokens

def \_auth\_headers() -> Dict[str, str]:

# Jira Cloud: Basic auth = base64(email:api\_token)

token\_bytes = f'{JIRA\_USER\_EMAIL}:{JIRA\_API\_TOKEN}'.encode('utf-8')

b64 = base64.b64encode(token\_bytes).decode('utf-8')

return {

'Authorization': f'Basic {b64}',

'Accept': 'application/json',

'Content-Type': 'application/json'

}

def \_get\_json(url: str, headers: Dict[str, str]) -> Any:

resp = requests.get(url, headers=headers)

if 200 <= resp.status\_code < 300:

return resp.json()

raise Exception(f'HTTP {resp.status\_code} for {url}: {resp.text}')

def \_build\_expand(expands: List[str]) -> str:

if not expands:

return ''

# Comma-separated list, e.g., expand=renderedFields,names,changelog

return 'expand=' + ','.join(expands)

def \_shape\_issue(issue: Dict[str, Any]) -> Dict[str, Any]:

# Keep the full raw issue but also extract a compact summary for quick use

fields = issue.get('fields', {})

project = fields.get('project') or {}

issuetype = fields.get('issuetype') or {}

status = fields.get('status') or {}

priority = fields.get('priority') or {}

reporter = fields.get('reporter') or {}

assignee = fields.get('assignee') or {}

parent = fields.get('parent') or {}

compact = {

'id': issue.get('id'),

'key': issue.get('key'),

'self': issue.get('self'),

'summary': fields.get('summary'),

'description': fields.get('description'),

'status': {'name': status.get('name'), 'id': status.get('id')},

'issuetype': {'name': issuetype.get('name'), 'id': issuetype.get('id')},

'priority': {'name': priority.get('name'), 'id': priority.get('id')} if priority else None,

'project': {'key': project.get('key'), 'name': project.get('name'), 'id': project.get('id')},

'reporter': {'accountId': reporter.get('accountId'), 'displayName': reporter.get('displayName')} if reporter else None,

'assignee': {'accountId': assignee.get('accountId'), 'displayName': assignee.get('displayName')} if assignee else None,

'parent': {'key': parent.get('key'), 'id': parent.get('id')} if parent else None,

'created': fields.get('created'),

'updated': fields.get('updated'),

'resolutiondate': fields.get('resolutiondate'),

'labels': fields.get('labels') or [],

'fixVersions': [{'name': fv.get('name')} for fv in (fields.get('fixVersions') or [])],

'components': [{'name': c.get('name')} for c in (fields.get('components') or [])]

}

shaped = {

'compact': compact,

'raw': issue # include the full raw issue for completeness

}

return shaped

def jira\_get\_issue\_details(

issue\_key: str,

expands: str = 'renderedFields,names,changelog,transitions,editmeta,versionedRepresentations' # adjust as needed

) -> str:

"""

Fetches a Jira issue with optional expansions.

Returns a JSON string with structure: {'issue': {'compact': {...}, 'raw': {...}}} or {'error': '...'}.

"""

if not issue\_key:

return json.dumps({'error': 'Missing issue\_key'})

if not (JIRA\_BASE\_URL and JIRA\_USER\_EMAIL and JIRA\_API\_TOKEN):

return json.dumps({'error': 'Missing Jira configuration (base URL, email, or API token).'})

headers = \_auth\_headers()

expand\_list = [e.strip() for e in expands.split(',') if e.strip()] if expands else []

expand\_qs = \_build\_expand(expand\_list)

url = f'{JIRA\_BASE\_URL}/rest/api/3/issue/{issue\_key}'

if expand\_qs:

url = f'{url}?{expand\_qs}'

try:

issue = \_get\_json(url, headers)

except Exception as e:

return json.dumps({'error': 'jira\_issue\_fetch\_failed', 'details': str(e)})

shaped = \_shape\_issue(issue)

# Optionally pull comments, attachments, worklogs via separate endpoints if not included by expands

# Comments (if not expanded via renderedFields/names):

# comments = \_get\_json(f'{JIRA\_BASE\_URL}/rest/api/3/issue/{issue\_key}/comment', headers)

# Attachments are within fields.attachment when available.

# Worklogs:

# worklogs = \_get\_json(f'{JIRA\_BASE\_URL}/rest/api/3/issue/{issue\_key}/worklog', headers)

return json.dumps({'issue': shaped}, ensure\_ascii=False)

if \_\_name\_\_ == "\_\_main\_\_":

# Example usage: replace with your own domain, email, token, and issue key

print(jira\_get\_issue\_details(issue\_key='PROJ-123'))