tasky_update_reminder

Description

Updates an existing Tasky reminder with new message, time, schedule days, or enabled status.

Purpose

Modify reminder properties to adjust notification schedules, update messages, or temporarily disable reminders. Supports partial updates - only specified fields are modified.

Parameters

Parameter	Type	Required	Description
`id`	string	✓	Reminder ID to update
`message`	string	✗	New reminder message text
`time`	string	✗	New time in HH:MM format
`days`	string[]	✗	New array of weekdays (replaces existing)
`enabled`	boolean	✗	Enable/disable the reminder

UI Flow

1. **User Input:** "Change my 9 AM reminder to 10 AM" or "Disable my weekend planning reminder"
2. **AI Processing:** Identifies reminder by ID/description and extracts update parameters
3. **Tool Call:** `mcpCall` invoked with reminder ID and updates
4. **Confirmation:** User sees confirmation overlay showing:
   - Current reminder details
   - Proposed changes highlighted
   - Impact on schedule (next occurrence change)
5. **Validation:** System verifies reminder exists and updates are valid
6. **Execution:** Upon approval, reminder updated in database
7. **Result Display:** Updated reminder details shown as adaptive card

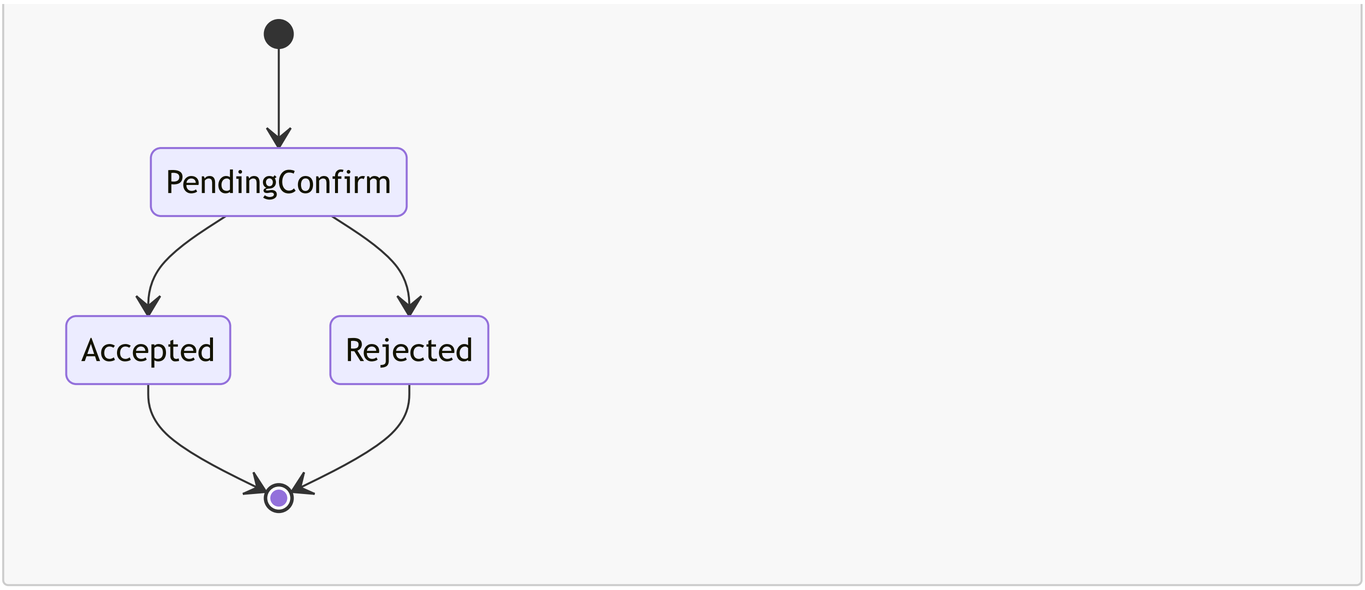
Database Operations

Confirmation Outcomes

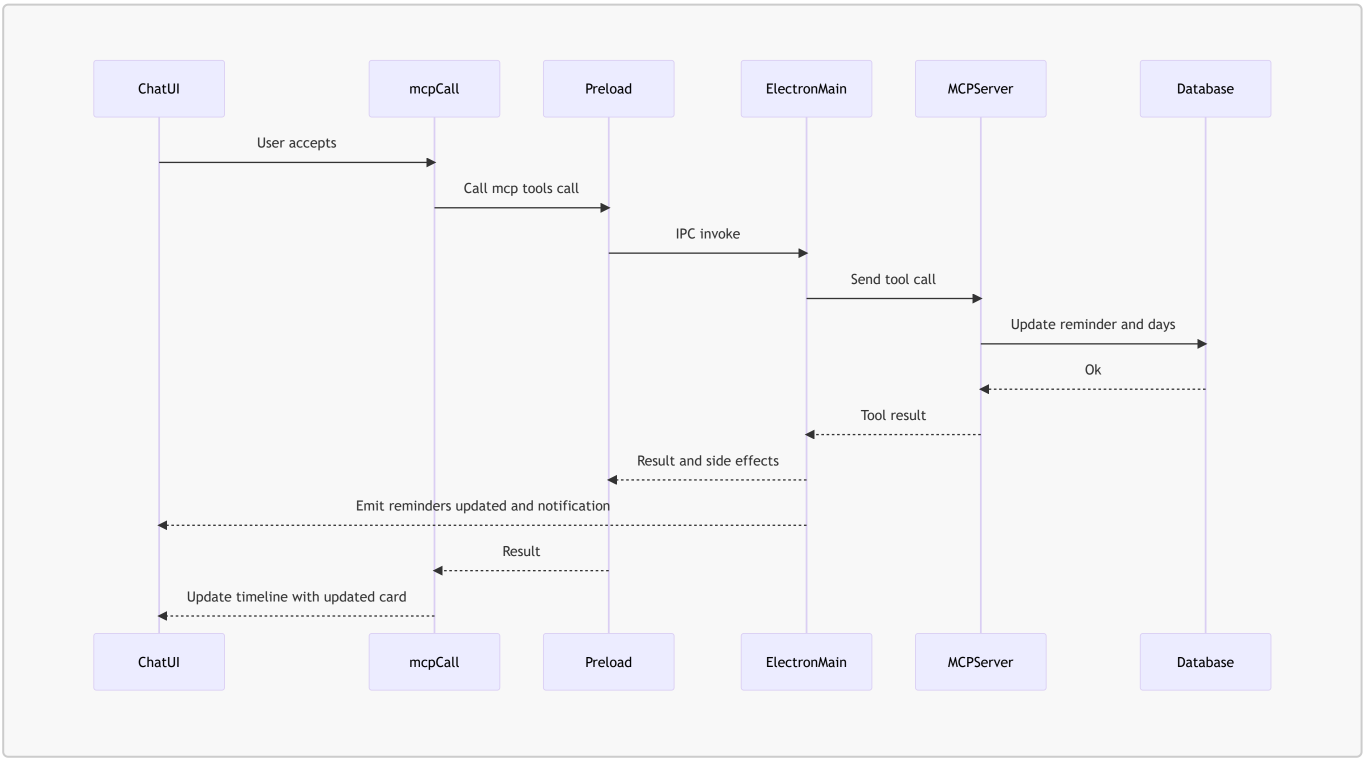
This tool requires user confirmation. Auto accept is not used for updates.

State

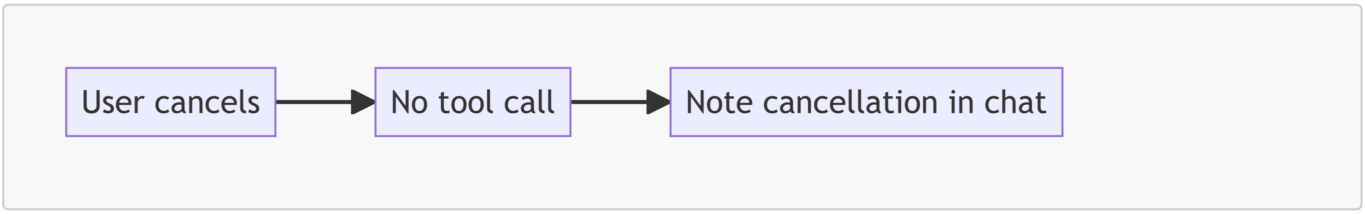
```

```



Accepted



Rejected



Auto accept

- Not applicable for update

Side effects on accept

- Emits `tasky:reminders-updated` event

- OS notification for updated reminder
- Adaptive card snapshot embedded in chat

See also: State Management Diagrams

Adaptive Card Response

Snapshot shape

```
{
  "__taskyCard": {
    "kind": "result",
    "tool": "tasky_update_reminder",
    "status": "success",
    "data": {
      "id": "abc123",
      "message": "Check emails",
      "time": "10:00",
      "days": ["monday","tuesday","wednesday","thursday","friday"],
      "enabled": true
    },
    "meta": { "operation": "update", "timestamp": "2025-09-17T16:00:00.000Z" }
  }
}
```

Error variant

```
{
  "__taskyCard": {
    "kind": "result",
    "tool": "tasky_update_reminder",
    "status": "error",
    "error": { "message": "Reminder not found", "code": "NOT_FOUND" }
  }
}
```

Renderer notes

- Success: Updated reminder card with changed fields emphasized.
- Error: Inline error with retry guidance.

```
-- Fetch current reminder for validation
SELECT * FROM reminders WHERE id = ?;

-- Update reminder with new values
UPDATE reminders SET
  message = @message,
  time = @time,
  enabled = @enabled,
  updated_at = @updated_at,
  metadata = @metadata
WHERE id = @id;

-- Update days (if provided) - replace existing
DELETE FROM reminder_days WHERE reminder_id = ?;
INSERT INTO reminder_days (reminder_id, day) VALUES (?, ?);
```

MCP Request Examples

Update Reminder Time

```
curl -X POST http://localhost:7845/mcp \
  -H "Content-Type: application/json" \
  -d '{
    "jsonrpc": "2.0",
    "id": 8,
    "method": "tools/call",
    "params": {
      "name": "tasky_update_reminder",
      "arguments": {
        "id": "reminder_20250907_164500_abc123",
        "time": "10:00"
      }
    }
  }'
```

Update Message and Days

```
curl -X POST http://localhost:7845/mcp \
  -H "Content-Type: application/json" \
  -d '{
    "jsonrpc": "2.0",
    "id": 9,
    "method": "tools/call",
    "params": {
      "name": "tasky_update_reminder",
      "arguments": {
        "id": "reminder_20250907_164500_abc123",
        "message": "Check emails and handle urgent requests",
        "days": ["monday", "tuesday", "wednesday", "thursday", "friday",
"saturday"]
      }
    }
  }'
```

Disable Reminder

```
curl -X POST http://localhost:7845/mcp \
  -H "Content-Type: application/json" \
  -d '{
    "jsonrpc": "2.0",
    "id": 10,
    "method": "tools/call",
    "params": {
      "name": "tasky_update_reminder",
      "arguments": {
        "id": "reminder_20250907_180000_ghi789",
        "enabled": false
      }
    }
  }'
```

Response Format

```
{
  "jsonrpc": "2.0",
  "id": 8,
  "result": {
    "content": [
      {
        "type": "text",
        "text": "Reminder updated successfully"
      },
      {
        "type": "text",
        "text": "{\"id\":\"reminder_20250907_164500_abc123\",\"message\":\"Check emails and respond to urgent items\",\"time\":\"10:00\",\"days\":[\"monday\",\"tuesday\",\"wednesday\",\"thursday\",\"friday\"],\"enabled\":true,\"oneTime\":false,\"createdAt\":\"2025-09-07T16:45:00.000Z\",\"updatedAt\":\"2025-09-07T18:30:00.000Z\",\"nextOccurrence\":\"2025-09-08T10:00:00.000Z\"}"
      }
    ]
  }
}
```

UI Components

- **ConfirmOverlay:** Shows update confirmation with:
  - Before/after comparison
  - Schedule changes highlighted
  - Next occurrence impact
  - Field-specific change summary
- **ToolCallDisplay:** Renders update operation status

- **AdaptiveCardRenderer:** Displays updated reminder with:
    - Modified fields highlighted
    - New schedule visualization
    - Status change indicators
    - Updated next occurrence time

Common Update Patterns

Time Changes

- **Later Start:** `{"time": "10:00"}` - Push back notification time
- **Earlier Start:** `{"time": "08:00"}` - Move up notification time
- **Lunch Time:** `{"time": "12:30"}` - Adjust for meal schedules

Schedule Changes

- **Add Weekend:** `{"days": ["monday", "tuesday", "wednesday", "thursday", "friday", "saturday", "sunday"]}`
- **Weekdays Only:** `{"days": ["monday", "tuesday", "wednesday", "thursday", "friday"]}`
- **Specific Days:** `{"days": ["monday", "wednesday", "friday"]}`

Content Updates

- **Clarify Message:** `{"message": "Check emails and respond to urgent requests"}`
- **Add Context:** `{"message": "Daily standup meeting - prepare updates"}`
- **Simplify:** `{"message": "Standup"}`

Status Management

- **Temporary Disable:** `{"enabled": false}` - Pause without deleting
- **Re-enable:** `{"enabled": true}` - Restore notifications
- **Vacation Mode:** Disable multiple reminders temporarily

Confirmation Dialog Examples

Time Change Confirmation

```
📝 Update Reminder

Current: Check emails at 9:00 AM (weekdays)
Change: Check emails at 10:00 AM (weekdays)

Next occurrence: Tomorrow at 10:00 AM (was 9:00 AM)

[Update Reminder] [Cancel]
```

Schedule Change Confirmation

```
📝 Update Reminder

Current: Weekend planning at 10:00 AM (Saturday)
Change: Weekend planning at 10:00 AM (Saturday, Sunday)

• Added Sunday to schedule
• Next occurrence: Tomorrow at 10:00 AM

[Update Reminder] [Cancel]
```

Disable Confirmation

```
📝 Update Reminder

Action: Disable "Daily standup meeting"
Current: Active (weekdays at 9:15 AM)
After: Disabled (no notifications)

This reminder will stop firing until re-enabled.

[Disable Reminder] [Cancel]
```

Validation Rules

Field	Validation
`id`	Must exist in database
`message`	Non-empty string if provided
`time`	Must be valid HH:MM format
`days`	Array of valid weekday strings
`enabled`	Boolean value

Error Handling

Error	Cause	Response
Reminder not found	Invalid `id` parameter	`{"content": [{"type": "text", "text": "Reminder not found"}], "isError": true}`
Missing ID	`id` parameter not provided	`{"content": [{"type": "text", "text": "id is required"}], "isError": true}`
Invalid time format	Malformed time string	Time parsing error with format example

Error	Cause	Response
Invalid days	Unknown weekday names	Validation error listing valid days
Database error	SQLite operation failure	Transaction rollback with error details

Next Occurrence Recalculation

When schedule or time changes, the system recalculates the next occurrence:

```
const recalculateNextOccurrence = (time: string, days: string[], enabled: boolean)
=> {
  if (!enabled) return null;

  const now = new Date();
  const [hours, minutes] = time.split(':').map(Number);

  // Find next matching day from updated schedule
  const nextMatchingDay = findNextDayInSchedule(days, now);
  nextMatchingDay.setHours(hours, minutes, 0, 0);

  // If next occurrence is in the past today, move to next cycle
  if (nextMatchingDay <= now) {
    return findNextCycle(days, nextMatchingDay);
  }

  return nextMatchingDay;
};
```

Partial Update Logic

The system supports partial updates, preserving unchanged fields:

```
const performUpdate = (currentReminder, updates) => {
  return {
    ...currentReminder,
    message: updates.message ?? currentReminder.message,
    time: updates.time ?? currentReminder.time,
    enabled: updates.enabled ?? currentReminder.enabled,
    days: updates.days ?? currentReminder.days,
    updatedAt: new Date(),
    nextOccurrence: recalculateNextOccurrence(
      updates.time ?? currentReminder.time,
      updates.days ?? currentReminder.days,
      updates.enabled ?? currentReminder.enabled
    )
  };
};
```

Schedule Impact Analysis

Updates that affect scheduling show impact analysis:

Change Type	Impact	User Notification
Time earlier	Next occurs sooner	"Next reminder moved up by X minutes"
Time later	Next occurs later	"Next reminder delayed by X minutes"
Days added	More frequent	"Added X days - reminder will fire more often"
Days removed	Less frequent	"Removed X days - reminder will fire less often"
Disabled	No notifications	"Reminder notifications paused"
Enabled	Resume notifications	"Reminder notifications resumed"

Transaction Safety

- **Atomic Updates:** Reminder and days updated in single transaction
- **Rollback:** Any failure rolls back all changes
- **Consistency:** Maintains referential integrity between tables
- **Validation:** Checks existence before update attempts

Performance Considerations

- **Single Query:** Fetches current reminder for validation
- **Partial Updates:** Only modified fields updated in SQL
- **Day Replacement:** Efficient delete-and-insert for days
- **Index Usage:** Benefits from primary key and time indexes
- **Schedule Calculation:** Next occurrence computed once

Integration with Notification System

Updated reminders immediately affect the notification scheduler:

- **Time Changes:** Next fire time updated in scheduler
- **Status Changes:** Enabled/disabled state synced
- **Schedule Changes:** New day pattern applied
- **Message Updates:** New content for future notifications

Related Components

- `tasky-mcp-agent/src/mcp-server.ts:268-295` - Tool definition and handler
- `tasky-mcp-agent/src/utils/reminder-bridge.ts` - Database update operations
- `src/components/chat/ConfirmOverlay.tsx` - Update confirmation UI
- `src/components/chat/AdaptiveCardRenderer.tsx` - Updated reminder display

Best Practices

1. **Clear Intent:** Confirm what's changing before update
2. **Schedule Awareness:** Consider impact on notification frequency
3. **Partial Updates:** Only change what needs changing
4. **Status Management:** Use disable instead of delete for temporary changes
5. **Time Zones:** Ensure time updates account for user timezone