PARTY PATTAYA BOT v10.1 -

46

`:`
`: @Party_Pattaya, +66-633-633-407, Liliya@partypattayacity.com`
`: 25.11.2025`
`: v10.1 FINAL COMPLETE`
`: PRODUCTION READY`
`: https://claude.ai/chat/acbf772e-c050-4502-b524-e6950bd7c233`

:

1. (greeting.json)
2. (@Party_Pattaya, +66-633-633-407, Liliya@partypattayacity.com)
3. (services.json)
4. (3 !)
5. (,)

:

- → →
- `~/Desktop/Bot Party Pattaya/blocks_ready/`
-
-
-

- 46

(1-10)

- 1: Registry & Recovery System
- 2: Configuration Manager
- 3: Database Manager
- 4: Cache System
- 5: Logging System
- 6: Error Handler
- 7: Security Manager
- 8: API Gateway
- 9: Rate Limiter
- 10: Health Monitor

(11-25)

- 11: User Manager
- 12: Session Manager
- 13: Command Router
- 14: Message Handler
- 15: Callback Handler

- 16: Voice Processor
- 17: Translation Engine
- 18: Media Manager
- 19: File Storage
- 20: Notification System
- 21: Booking Manager
- 22: Payment Processor
- 23: Invoice Generator
- 24: Calendar Manager
- 25: Review System

AI (26-35)

- 26: AI Planning Agent
- 27: AI Code Generator
- 28: AI Validator
- 29: AI Optimizer
- 30: AI Documentation
- 31: AI Monitoring
- 32: AI Billing
- 33: AI Analytics
- 34: Chatbot Engine
- 35: Smart Recommendations

(36-46)

- 36: Universal Protection System
- 37: Telegram Integration
- 38: WhatsApp Integration
- 39: Email Integration
- 40: Social Media Manager
- 41: CRM Integration
- 42: Analytics Dashboard
- 43: Reporting System
- 44: Backup Manager
- 45: Update Manager
- 46: Admin Panel

1: REGISTRY & RECOVERY SYSTEM

:

: `blocks_ready/block_01_registry.py`
: `36KB (~900 )`
: `1.0`
: `PRODUCTION READY`
: `watchdog-6.0.0`

- . , ,
(Google Drive, Telegram, Claude Chat).

19 (280%)

(1-5)

1. register_block()

```
def register_block(
    block_id: int,
    block_name: str,
    file_path: str,
    status: str = 'development',
    version: str = '1.0',
    dependencies: list = None,
    chat_url: str = None,
    gdrive_url: str = None,
    description: str = None
) -> bool:
    """



    Args:
        block_id:         (1-46)
        block_name:
        file_path:
        status:      (development/testing/ready/production/deprecated/paused)
        version:      (    : MAJOR.MINOR.PATCH)
        dependencies:                [1, 2, 3]
        chat_url:           Claude
        gdrive_url:        Google Drive
        description:

    Returns:
        True

    Example:
        register_block(
            block_id=1,
            block_name='Registry System',
            file_path='blocks_ready/block_01_registry.py',
            status='ready',
            version='1.0',
            dependencies=[],
            chat_url='https://claude.ai/chat/abc123',
            description='               '
        )
    """
```

: - SHA256 - - JSON - Timestamp - (1-46) - - (semver)

2. recover_block()

```
def recover_block(
    block_id: int,
    source: str = 'auto'
) -> bool:
    """


    Args:
        block_id:
        source:
            - 'auto':
            - 'registry':
            - 'chat':   Claude
            - 'gdrive':   Google Drive
            - 'backup':          backup

    Returns:
        True

    Example:
        #
        recover_block(1, 'auto')

        #
        recover_block(36, 'gdrive')
    """
```

: 1. **auto** : - registry → chat → gdrive → backup - 2. **registry:** block_registry.json 3. **chat:** Claude 4. **gdrive:** Google Drive 5. **backup:** backups/

3. update_block_version()

```
def update_block_version(
    block_id: int,
    update_type: str = 'patch'
) -> str:
    """
                 (                  )

    Args:
        block_id:
        update_type:
            - 'major': 1.0.0 → 2.0.0 (breaking changes)
            - 'minor': 1.0.0 → 1.1.0 (new features)
            - 'patch': 1.0.0 → 1.0.1 (bug fixes)

    Returns:


    Example:
        update_block_version(1, 'minor')  # 1.0 → 1.1
        update_block_version(36, 'major')  # 1.0 → 2.0
    """
```

: - **MAJOR:** API - **MINOR:** , - **PATCH:**

4. save_chat_link()

```python
def save_chat_link(
    block_id: int,
    chat_url: str
) -> bool:
    """
                    Claude

    Args:
        block_id:
        chat_url: URL     Claude

    Returns:
        True

    Example:
        save_chat_link(1, 'https://claude.ai/chat/abc123')
    """
```

: -   URL -   claude.ai -

5. list_all_blocks()

```python
def list_all_blocks(
    status_filter: str = None,
    show_dependencies: bool = False
) -> None:
    """


    Args:
        status_filter:             (None =   )
        show_dependencies:

    Example:
        list_all_blocks()  #
        list_all_blocks('ready')  #
        list_all_blocks(show_dependencies=True)  #
    """
```

: - ready - development - testing - production - deprecated - paused

(6-10)

6. export_chat_text()

```python
def export_chat_text(
    block_id: int,
    chat_text: str
) -> str:
    """


    Args:
        block_id:
        chat_text:
```

```
    Returns:


    Example:
        path = export_chat_text(1, "       ...")
    """
```

: - : docs/chat_history/ - : block_{id:02d}_chat_{timestamp}.txt - Markdown

7. save_gdrive_link()

```
def save_gdrive_link(
    block_id: int,
    gdrive_url: str
) -> bool:
    """
                Google Drive

    Args:
        block_id:
        gdrive_url: URL Google Drive

    Returns:
        True
    """
```

8. get_registry_report()

```
def get_registry_report(
    format: str = 'detailed'
) -> str:
    """



    Args:
        format:
            - 'detailed':
            - 'simple':
            - 'stats':

    Returns:


    Example:
        report = get_registry_report('detailed')
        print(report)
    """
```

: - - - - -

9. register_inter_block_api()

```
def register_inter_block_api(
    from_block: int,
    to_block: int,
    api_method: str,
    description: str = None
) -> bool:
    """
           API

    Args:
        from_block: ID    -
        to_block: ID    -
        api_method:           API
        description:

    Returns:
        True

    Example:
        register_inter_block_api(
            from_block=1,
            to_block=36,
            api_method='get_block_status',
            description='                    '
        )
    """
```

: -   -   -

10. notify_dependent_blocks()

```
def notify_dependent_blocks(
    block_id: int,
    message: str
) -> list:
    """


    Args:
        block_id: ID
        message:

    Returns:


    Example:
        notify_dependent_blocks(1, "            1.1")
    """
```

(11-15)

11. start_file_monitoring()

```
def start_file_monitoring(
    watch_directory: str = 'blocks_ready'
) -> None:
    """
                        watchdog
    Args:
        watch_directory:

    Example:
        #
        import threading
        monitor = threading.Thread(
            target=start_file_monitoring,
            daemon=True
        )
        monitor.start()
    """
```

: - → - → SHA256 - → deprecated - →

12. cli_menu()

```
def cli_menu() -> None:
    """
            CLI

    13    :
    1.
    2.
    3.
    4.
    5.
    6.              Google Drive
    7.
    8.        backup
    9.
    10.
    11. Health check
    12. Batch
    13.

    Example:
        cli_menu()
    """
```

13. rollback_to_version()

```
def rollback_to_version(
    block_id: int,
    target_version: str
) -> bool:
    """


    Args:
        block_id: ID
        target_version:

    Returns:
        True

    Example:
        rollback_to_version(1, '1.0.0')
    """
```

: - backups/ - -

14. analyze_dependencies()

```
def analyze_dependencies() -> dict:
    """


    Returns:
        {
            'graph':           ,
            'cycles':               ,
            'orphans':               ,
            'critical':
        }

    Example:
        analysis = analyze_dependencies()
        print(f"   : {len(analysis['cycles'])}")
    """
```

15. health_check_all_blocks()

```
def health_check_all_blocks() -> dict:
    """


    Returns:
        {
            'healthy':               ,
            'corrupted':             (SHA256        ),
            'missing':                  ,
            'outdated':
        }

    Example:
        health = health_check_all_blocks()
        if health['corrupted']:
            print(f"          : {health['corrupted']}")
    """
```

(16-19)

16. register_all_blocks_in_directory()

```
def register_all_blocks_in_directory(
    directory: str = 'blocks_ready',
    auto_status: str = 'development'
) -> int:
    """
    Batch

    Args:
        directory:
        auto_status:

    Returns:


    Example:
        count = register_all_blocks_in_directory('blocks_ready')
        print(f"         : {count}")
    """
```

17. search_blocks()

```
def search_blocks(
    query: str = None,
    status: str = None,
    has_dependencies: bool = None,
    version_min: str = None
) -> list:
    """


    Args:
        query:           (    ,     )
        status:
        has_dependencies:      /
        version_min:

    Returns:


    Example:
        #
        blocks = search_blocks(
            status='ready',
            has_dependencies=True
        )
    """
```

18. send_telegram_notification()

```
def send_telegram_notification(
    message: str,
    priority: str = 'normal'
) -> bool:
    """
                 Telegram

    Args:
        message:
        priority:
            - 'low':
```

```
            - 'normal':
            - 'high':
            - 'critical':

    Returns:
        True

    Example:
        send_telegram_notification(
            "   36       !",
            priority='critical'
        )
    """
```

Telegram : - Bot Token: .env - Chat ID: Admin ID (359364877) - : Markdown - Emoji

19. export_registry() / import_from_backup()

```
def export_registry(
    format: str = 'json',
    output_path: str = None
) -> str:
    """


    Args:
        format:           (json/csv/markdown)
        output_path:

    Returns:


    Example:
        export_registry('markdown', 'docs/registry_report.md')
    """

def import_from_backup(
    backup_path: str
) -> bool:
    """
               backup

    Args:
        backup_path:      backup

    Returns:
        True
    """
```

block_registry.json

```json
{
  "blocks": {
    "1": {
      "block_id": 1,
      "block_name": "Registry & Recovery System",
      "file_path": "blocks_ready/block_01_registry.py",
      "status": "ready",
      "version": "1.0",
      "dependencies": [],
      "hash": "abc123...def456",
      "created_at": "2025-11-25T10:00:00",
      "updated_at": "2025-11-25T15:30:00",
      "chat_url": "https://claude.ai/chat/abc123",
      "gdrive_url": null,
      "description": "               "
    },
    "36": {
      "block_id": 36,
      "block_name": "Universal Protection System",
      "file_path": "blocks_ready/block_36_protection.py",
      "status": "ready",
      "version": "1.0",
      "dependencies": [1],
      "hash": "xyz789...uvw012",
      "created_at": "2025-11-25T12:00:00",
      "updated_at": "2025-11-25T16:00:00",
      "chat_url": "https://claude.ai/chat/abc123",
      "gdrive_url": null,
      "description": "               "
    }
  },
  "inter_block_apis": [
    {
      "from_block": 1,
      "to_block": 36,
      "api_method": "get_block_status",
      "description": "              "
    }
  ],
  "metadata": {
    "total_blocks": 2,
    "last_updated": "2025-11-25T16:00:00",
    "version": "1.0"
  }
}
```

36: UNIVERSAL PROTECTION SYSTEM

:

: `blocks_ready/block_36_protection.py`
: `13KB`
: `1.0`
: `PRODUCTION READY`
: `watchdog`

. SHA256 , , backup/restore Telegram .

13

(1-5)

1. calculate_hash()

```
def calculate_hash(file_path: Path) -> str:
    """
          SHA256

    Args:
        file_path:

    Returns:
        SHA256      hex
    """
```

2. save_hashes()

```
def save_hashes(hashes: dict) -> bool:
    """


    File: protected_hashes.json
    """
```

3. check_integrity()

```
def check_integrity() -> tuple[bool, list]:
    """


    Returns:
        (all_ok, list_of_modified_files)
    """
```

4. create_backup()

```
def create_backup(
    file_path: Path,
    backup_dir: Path = Path('backups')
) -> Path:
    """
         backup

    Format: {filename}_{timestamp}.bak
    """
```

5. restore_from_backup()

```
def restore_from_backup(
    file_name: str,
    backup_file: str = 'latest'
) -> bool:
    """
              backup
    """
```

(6-8)

6. log_message()

```
def log_message(
    level: str,
    message: str,
    extra: dict = None
) -> None:
    """


    File: logs/protection.log
    """
```

7. send_telegram_alert()

```
async def send_telegram_alert(
    message: str,
    priority: str = 'normal',
    admin_id: int = 359364877
) -> bool:
    """
                 Telegram

    Priorities: low, normal, high, critical
    """
```

8. discover_all_blocks()

```
def discover_all_blocks() -> dict:
    """


    Returns:
        {filename: Path}
    """
```

(9-13)

9. protect_python_blocks()

```
def protect_python_blocks(
    blocks_dir: Path = Path('blocks_ready')
) -> int:
    """
           Python

    Pattern: block_*.py
    """
```

10. integrate_with_registry()

```
def integrate_with_registry() -> bool:
    """
                1 (Registry)
    """
```

11. auto_restore_corrupted_files()

```
async def auto_restore_corrupted_files(
    notify: bool = True
) -> list:
    """

    """
```

12. full_setup()

```
def full_setup() -> dict:
    """

    """
```

13. show_status()

```
def show_status(detailed: bool = False) -> dict:
    """

    """
```

JSON (5)

```
greeting.json    -            (     !)
contacts.json    -       (     !)
services.json    -          (     !)
buttons.json     -     3    ! (     !)
tz_v10_1.json    -             (     !)
```

Python (46)

```
block_01.py - block_46.py
```

2-46:

2: Configuration Manager

-
- Pydantic
-
-
- 12

3: Database Manager

- PostgreSQL
- Connection pooling
-
- Backup/restore
- 15

4: Cache System

- Redis
-
- TTL management
- Rate limiting support
- 11

5: Logging System

-
-
- Rotation
- JSON
- 10

6: Error Handler

-
- Retry
- Graceful degradation
- Error reporting
- 8

7: Security Manager

-
-
- Rate limiting
- XSS/CSRF
- 12

8: API Gateway

-
- Request/Response logging
-

- Load balancing
- 10

9: Rate Limiter

- Token bucket
- Per-user/IP
-
- Redis backend
- 7

10: Health Monitor

-
- Health checks
- Metrics collection
- Alerting
- 9

11: User Manager

- CRUD
-
- Preferences
- Analytics
- 14

12: Session Manager

-
- State management
- Context preservation
- Cleanup
- 10

13: Command Router

-
- Parsing
- Middleware
- Error handling
- 8

14: Message Handler

-
- Queuing
- Priority
- Batch processing
- 11

15: Callback Handler

- Callback queries
- State machines

- Inline keyboards
- Action routing
- 9

16: Voice Processor

- Whisper STT
- OpenAI TTS
- Format conversion
- Quality optimization
- 12

17: Translation Engine

- 100+
- Auto-detection
- Context-aware
- Caching
- 10

18: Media Manager

- File upload
- Image processing
- Video processing
- Storage optimization
- 13

19: File Storage

- S3/Local storage
- CDN integration
- Compression
- Cleanup policies
- 11

20: Notification System

- Push notifications
- Email
- SMS
- Scheduling
- 10

21: Booking Manager

- Service booking
- Availability
- Confirmation
- Cancellation
- 15

22: Payment Processor

- Multiple gateways

- Recurring payments
- Refunds
- Webhooks
- 14

23: Invoice Generator

- PDF invoices
- Email delivery
- Payment tracking
- Tax calculations
- 10

24: Calendar Manager

- Booking calendar
- Availability
- Timezone handling
- Conflict detection
- 12

25: Review System

- Ratings & reviews
- Moderation
- Analytics
- Display widgets
- 11

26: AI Planning Agent

- Task planning
- Priority assignment
- Resource allocation
- Timeline estimation
- 10

27: AI Code Generator

- Code generation
- Template engine
- Validation
- Version control
- 12

28: AI Validator

- Code validation
- Security checks
- Performance
- Best practices
- 9

29: AI Optimizer

- Code optimization
- Query optimization
- Cache strategies
- Performance tuning
- 11

30: AI Documentation

- Auto-generated docs
- API documentation
- Code comments
- User guides
- 10

31: AI Monitoring

- Anomaly detection
- Predictive maintenance
- Performance prediction
- Auto-scaling
- 13

32: AI Billing

- Usage tracking
- Cost optimization
- Budget alerts
- Invoice generation
- 10

33: AI Analytics

- Behavioral analysis
- Trend prediction
- Churn prediction
- Recommendations
- 14

34: Chatbot Engine

- Conversational AI
- Intent recognition
- Context management
- Multi-turn dialogues
- 12

35: Smart Recommendations

- Personalized suggestions
- Collaborative filtering
- Content-based
- A/B testing
- 11

37: Telegram Integration

- Bot API wrapper
- Webhook handling
- Media handling
- Payment integration
- 16

38: WhatsApp Integration

- Business API
- Templates
- Media sending
- Status tracking
- 12

39: Email Integration

- SMTP/IMAP
- Template engine
- Attachments
- Tracking
- 10

40: Social Media Manager

- Multi-platform
- Scheduling
- Analytics
- Engagement
- 13

41: CRM Integration

- Customer sync
- Lead management
- Sales pipeline
- Reporting
- 14

42: Analytics Dashboard

- Real-time metrics
- Custom reports
- Visualization
- Export
- 15

43: Reporting System

- Automated reports
- Custom templates
- Scheduling
- Distribution
- 11

44: Backup Manager

- Automated backups
- Incremental
- Restore procedures
- Verification
- 10

45: Update Manager

- Version management
- Rolling updates
- Rollback
- Zero-downtime
- 9

46: Admin Panel

- Web interface
- User management
- Configuration
- Monitoring dashboard
- 18

---

greeting.json

```json
{
  "text": "             Party Pattaya!\n\n                              :\n\n       \n              \n V
  "buttons": ["    ", "       ", "  VIP"]
}
```

contacts.json

```json
{
  "telegram": "@Party_Pattaya",
  "whatsapp": "+66-633-633-407",
  "email": "Liliya@partypattayacity.com",
  "admin_id": 359364877
}
```

services.json

```json
{
  "yachts": {
    "min_price": 500,
    "max_price": 2000,
    "currency": "USD"
  },
  "parties": {
    "min_price": 1000,
    "max_price": 5000,
    "currency": "USD"
  },
  "vip": {
    "min_price": 2000,
    "max_price": 10000,
    "currency": "USD"
  },
  "transfers": {
    "min_price": 20,
    "max_price": 200,
    "currency": "USD"
  }
}
```

buttons.json

```
{
  "main_menu": ["    ", "       ", "  VIP"],
  "max_buttons": 3,
  "note": "    3    !"
}
```

```
OpenAI API: $150/
Google Cloud: $50/
Hosting: $100/
Domain & SSL: $1/
   : $301/    ($3,612/  )
```

2026

```
Nov 2025 - Dec 2026: 14
$301 × 14 = $4,214
```

1: (1-2)

- 1: Registry
- 2-10

2: (3-4)

- 11-25

3: AI (5-6)

- 26-35

4: (7-8)

- 36: Protection
- 37-46

```
    : 2/46      (4.3%)
     : 44     (95.7%)
    : ~49KB
    : ~1,200
  : 32
         : 51
```

- : https://claude.ai/chat/acbf772e-c050-4502-b524-e6950bd7c233
- :
- Telegram: @Party_Pattaya
- WhatsApp: +66-633-633-407
- Email: Liliya@partypattayacity.com
- Bot Token: 8526699649:AAHKQN_HRkvMGcto7rrljdbsLPiGTGovYJY
- Admin ID: 359364877

---

1. :
    -
    -
    -
    - (3!)
    -
2. :
    - → →
    -
    -
3. :
    - 36
    -
    - Telegram

: v10.1 FINAL COMPLETE
: 25.11.2025
: 131KB
: 46/46
: (@Party_Pattaya)